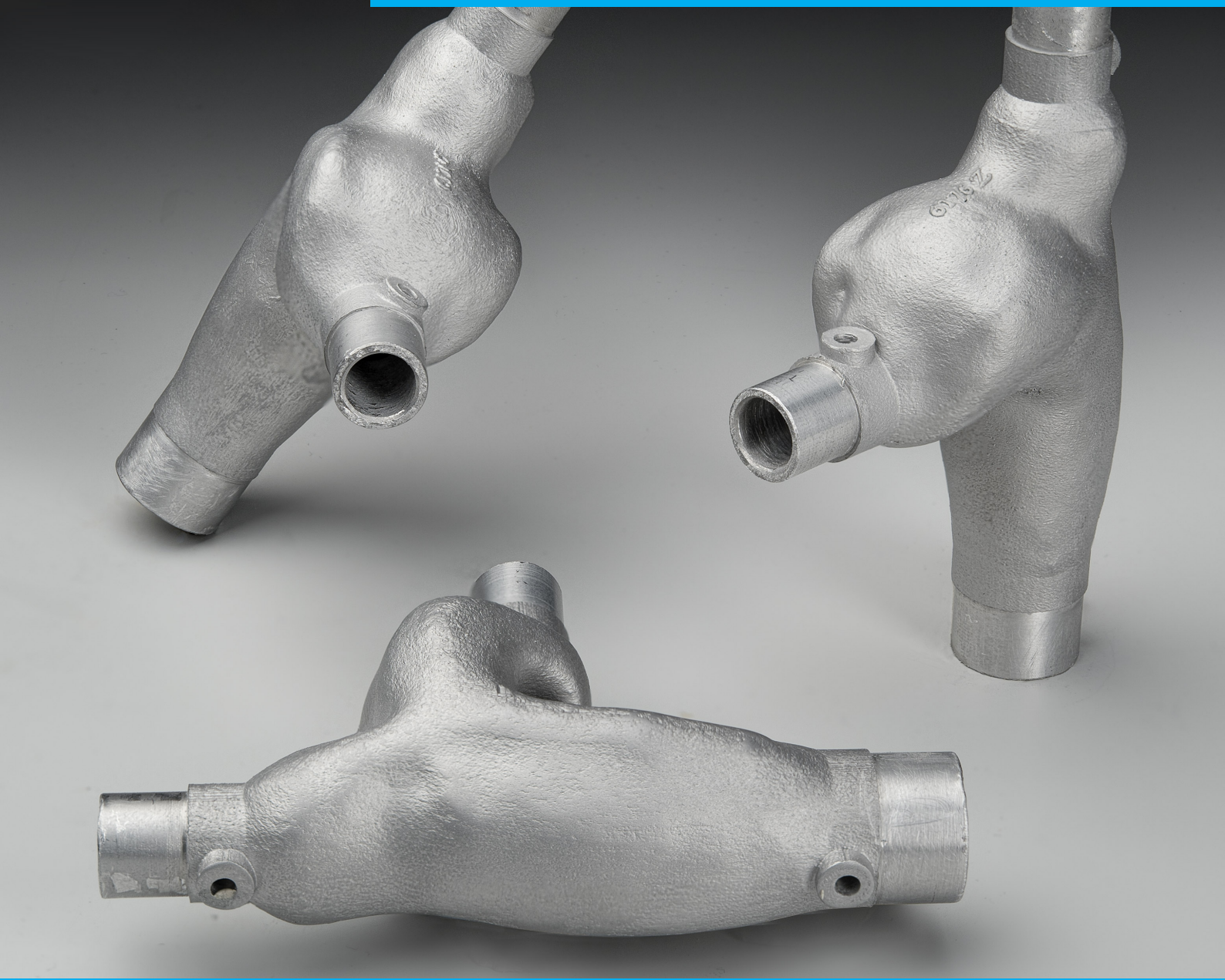
Department of Precision and Microsystems Engineering

Design and Additive Manufacturing of Manifolds for Navier-Stokes Flow: A Topology Optimisation Approach

J.M. Verboom

Report number : 2017.025
Coaches : MSc. E.A. van de Ven, MSc. W. van den Brink
Professor : Dr. ir. M. Langelaar
Specialisation : Engineering Mechanics
Type of report : Master Thesis
Date : 22-06-2017



TU Delft

DESIGN AND ADDITIVE MANUFACTURING OF MANIFOLDS FOR NAVIER-STOKES FLOW: A TOPOLOGY OPTIMISATION APPROACH

DESIGN AND ADDITIVE MANUFACTURING OF MANIFOLDS FOR NAVIER-STOKES FLOW: A TOPOLOGY OPTIMISATION APPROACH

Master Thesis

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 22, 2017 at 14:00.

by

J. M. VERBOOM

Faculty of Mechanical Engineering,
Delft University of Technology, Delft, The Netherlands

Student:

J. M. Verboom,	Delft University of Technology
Student number,	4290224
Project duration,	September 1, 2016 – June 22, 2017

Thesis committee:

Dr. ir. M. Langelaar,	Delft University of Technology
Dr. ir. R.A.J. van Ostayen,	Delft University of Technology
MSc. E.A. van de Ven,	Delft University of Technology
MSc. W. van den Brink,	Dutch Aerospace Laboratory (NLR)

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

"If I have seen further,
it is by standing on the shoulders of giants."

Isaac Newton

SUMMARY

The Netherlands Aerospace Centre is increasingly improving Selective Laser Melting (SLM) capabilities. It is working with academics and industry on demonstrators to prove the current state of the art of SLM. Recently they started with additive manufacturing (AM) of internal manifolds. This thesis aims to provide a novel design approach by including topology optimisation for Navier-Stokes flow with the focus on SLM. This approach intents to bridge the gap between academics and industry regarding fluid topology optimisation by practically implementing the methodology. As a result, this thesis places emphasis on 3D topology optimisation instead of the very common 2D approach in academics.

Fluid topology optimisation was chosen as the primary design step because it is a relatively new and unexplored method in academics and especially new in the industry. Moreover, it is a highly versatile design approach as it can come with any shape that yields the best result for the given objective and boundary conditions. Due to the complex nature of topology optimisation and its numerical implementation, it was chosen to divide this topology optimisation approach into multiple steps. Hence the goal of this thesis is to combine AM and topology optimisation of Navier-Stokes flow and to incorporate it into a design approach. The flow optimisation step aims to improve the performance, that is, minimising the power dissipation of the problem at hand. Besides incorporating fluid optimisation, this method also includes a weight reduction step by introducing constraints (volume and perimeter constraints), a specific AM post processing step to create self-supporting cross sections and a mass minimising structural topology optimisation step. Combining these methods lay the foundation for this unique design process for additive manufacturing.

The volume constraint is not directly proportional to the weight of the manifold, but it is an intuitive constraint that penalises the amount of fluid domain that can be used. The second constraint that is implemented is the perimeter constraint (including a greyness constraint). This constraint penalises the perimeter of a manifold, which is directly proportional to the weight of the manifold. The novelty of the perimeter constraint is that in fluid flow optimisation it serves as a direct mass control of the structural part of our manifold. Including this constraint will result in rather straight manifolds as opposed to the more organic shapes arising from the volume constraint.

A three-dimensional manifold is used as the case study for thesis and is compared to a manifold that would be manufactured conventionally. The performance objective is power dissipation minimization, and it is shown to reduce the power dissipation by 47% as compared to the conventional tee, almost doubling the performance.

The optimised manifolds are not always manufacturable due to the limitations of the AM process. In particular downward facing surfaces perpendicular to the build direction (overhang) cannot be printed without support. Adding sacrificial support material is not an option in manifolds, as it often cannot be removed. Therefore, a post-processing step is implemented to adapt the geometries to remove any internal overhanging regions. This post process step results in self-supporting manifold cross sections that are shaped like a droplet i.e. a 45-degree roof on top of a circular cross section. This approach yields manufacturable design, but it comes at the cost of performance. The x, y, and z build direction adjustments decreased the power dissipation by 41.5%, 27.9%, and 38.4%, respectively. Each design adjustment reduces the performance of the tee; nonetheless, they perform significantly better than the conventional tee.

Subsequently, a method to optimise the local wall thickness is applied. For this purpose, the internal pressure field can be used which is found with the computational fluid dynamics (CFD) simulation . This pressure field acts as the load condition in the structural topology optimisation. The result from this approach is a progressive wall thickness such that it is thick where pressures are high, and thin where pressures are low.

The resulting geometries were manufactured after executing the completed design approach. It is shown that this method works well and can be practically implemented. A pressure measurement test showed a difference of 2.4% to 11.8% as compared to the simulations for the different designs, which is reasonable considering various assumptions such as the frozen turbulence, the turbulence model, mesh size and wall functions. Another test is carried out to look at the flow distribution of the tee and showed a larger difference of 2.8%-24% in flow distribution ratios. This larger difference is likely the effect of transient effects and possible internal roughness in the topology optimised manifold.

ACKNOWLEDGEMENTS

I would like to thank my supervisors from the Dutch Aerospace Laboratory (NLR), Wouter van den Brink and Marc de Smit, for their support during the last nine months. The freedom and responsibility for my research were very motivating. Also, I would like to thank many colleagues for their patience when I had lots of questions namely: Aswin Pauw, Ralph Haagsma, Bernard Bosma, Marcel van der Kroeg and Henk-Jan van Gerner. Lastly, I would like to thank everyone at NLR for a great experience and a pleasant working environment.

At the Delft University of Technology, I would like to thank Matthijs Langelaar for supervising me during this project. The feedback during our meetings was very useful and helped me improve my research. Also, my daily supervisor, Emiel van de Ven, was always enthusiastic to discuss and help with any problem. His ongoing support and dedication helped me to push forward during this project. Therefore special thanks to Emiel.

Above all I would like to thank my parents for their ongoing support to continue studying, my brother and sister for their guidance through the years and of course my girlfriend for her support and patience during this long but rewarding journey.

Jeroen Verboom
Delft, June 2017

Contents

List of Abbreviations

CFD	Computational Fluid Dynamics
NS	Navier-Stokes
RANS	Reynolds Avaraged Navier Stokes
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
PDE	Partial Differential Equation
OF	OpenFOAM (C++ CFD library)
TO	Topology Optimisation
SIMP	Solid Isotropic Material with Penalization
AM	Additive Manufacturing
SLM	Selective Laser Melting
SLS	Selective Laser Sintering
FDM	Fused Deposition Modeling

List of Symbols

α	Porosity design variable [-]
α_i^c	Current porosity field, element i
α_i^p	Printable porosity field, element i
α_j^p	Printable porosity field, element j = 1,2,..,9
$\delta\theta$	Shear strain angle of a fluid element [°]
Δp	Pressure difference $[Pa]$
δ_{ij}	Kronecker Delta [-]
ϵ	Turbulent Dissipation rate $[\frac{m^2}{s^3}]$
$\frac{\partial c_{grey}}{\partial \alpha}$	Greyness constraint derivative [-]
$\frac{\partial c_{perim}}{\partial \alpha}$	Perimeter constraint derivative [-]
$\frac{\partial L}{\partial \alpha_i}$	Sensitivity with respect to the design variable [-]
λ_{vol}	Volume constraint Lagrange multiplier [-]
$\mathbf{R}$	State equations [-]
$\mathbf{u}$	Adjoint Velocity [-]
$\mathbf{v}$	Velocity field $[\frac{m}{s}]$
$\mathbf{v}^d$	Target velocity (User defined) $[\frac{m}{s}]$
∇	Gradient operator [-]
ν	Dynamic viscosity $[\frac{kg}{ms}]$
ν_T	Turbulent viscosity $[Pa \cdot s]$
ρ	Density $[\frac{kg}{m^3}]$
τ	Shear stress $[Pa]$
A	Surface of cross section $[m^2]$
C_ν	Eddy viscosity constant [-]
c_{grey}	Greyness constraint [-]
c_{perim}	Perimeter constraint [-]
c_{vol}	Volume constraint [-]
D	Hydraulic diameter $[m]$
$D(\mathbf{u})$	Adjoint rate of strain tensor [-]
$D(\mathbf{v})$	Rate of strain tensor [-]
$d\partial\Omega_i$	Surface of the inlet and/or outlet [-]
Da	Darcy number [-]
H	Mean curvature [-]

I	Turbulent Intensity [-]
J	Objective Function [-]
J_{pl}	Power dissipation objective function $[W]$
J_{uni}	Objective function for uniform flow at outlet $[\frac{m}{s}]$
k	Turbulent Kinetic energy $[\frac{m^2}{s^2}]$
L	Augmented Lagrange Function [-]
L	Augmented Lagrange Function [-]
n_i	Surface normal [-]
p	Pressure field $[Pa]$
Q	Volume flow $[\frac{m^3}{s}]$
q	Adjoint Pressure [-]
Re	Reynolds Number [-]
V_{target}	User defined target volume [%]
w_{vol}	Volume constraint weight factor [-]

1

INTRODUCTION

In the aerospace sector, many systems still work based on hydraulics. Common examples of hydraulic systems are the flaps and landing gears of an aircraft or hydraulic systems in rockets like the grid fins of the Falcon 9 first stage and much more. For a commercial aircraft, the lifetime cost per kg is about 2000$/kg [36] for gross mass. However, if weight reductions are achieved early in the design phase, it will affect fuel savings, smaller engines, and smaller wings. Therefore the net effect of reducing mass is not only measurable by lifetime cost per kg. In the space sector, the gains from mass reduction are even higher and a financial but common metric to show how important weight is, is to look at the price per kg to low earth orbit. In the space shuttle era, the price per kg to low earth orbit was estimated to be $18.000/kg. However, some of the worst case estimations are up to $60.000/kg [59]. Fortunately with new commercial companies competing for the price has dropped to about $4.000/kg for the Falcon 9 from SpaceX, currently one of the cheapest rockets available. However, it is still a significant cost factor and crucial to reducing mass when possible.

Mainly two improvements have been identified; if the mass of components is reduced, more mass is available for the intended goal of aircraft and rockets, that is, carrying passengers or satellites, respectively. These improvements will have a direct impact on the cost per kg in the aerospace sector. Secondly, there is an indirect effect if the performance of hydraulic parts is improved regarding power dissipation. Reducing the power dissipation allows the designer to reduce the size of accompanied systems like pumps.

To tackle these two problems, weight and performance, the focus of this thesis is to increase the performance of fluid flow manifolds while reducing the weight. For this purpose Additive manufacturing (AM) and topology optimisation (TO) are identified as a potent combination to tackle mass reduction. Besides the advantage of reducing mass, this combination of methods can also increase the performance of parts by using material where it matters most by changing the shape of a manifold to improve the efficiency of the fluid flow. Optimised designs obtained by topology optimisation are often of such complexity, that additive AM becomes essential for fabrication.

This research is a collaboration between the Netherlands Aerospace Centre (NLR) and the Delft University of Technology. The NLR is dedicated to innovation in aerospace, and this thesis aims to contribute to their mission. By using both AM and TO, we seek to identify a design approach to improve manifolds by minimising power dissipation and mass.

1

1.1. RESEARCH GOALS

The goal of this project is to investigate the design approach from topology optimisation of fluid flow manifolds to the manufacturing process with Selective Laser Melting (SLM). During the design process two criteria ought to be optimised; power dissipation and the mass of a manifold. This can be summarised in several questions for the literature survey and the main research.

1.1.1. LITERATURE SURVEY

During the literature survey, several research questions were proposed which serve as the foundation for this thesis.

- What are the current capabilities of Selective Laser Melting (SLM) for internal manifolds?
- What are the benefits and drawbacks of manufacturing manifolds with SLM?
- What method is currently available for fluid flow topology optimisation that is useful for practical 3D implementation?
- What software packages are available for fluid flow optimisation and usable for this research?

Furthermore, fluid mechanics topics have been reviewed that are necessary to complete this research successfully. It should be noted that this research is not focused on fluid mechanics or computational fluid dynamics (CFD). However, it is necessary that the important concepts are explained and understood.

1.1.2. RESEARCH

The main research revolves around the design approach, starting from the design domain through the manufacturing process with Selective Laser Melting. This thesis aims to provide a general methodology to go from a design domain to an optimised design that is well suited for AM. The following five questions give a broad overview of the design approach which is implemented throughout this thesis.

- Can the performance (power dissipation) of manifolds be improved with continuous adjoint topology optimisation?
- What methods can be used to minimise the mass of manifolds manufactured with SLM?
- How can the geometry of internal manifolds be adapted to meet SLM manufacturing requirements and how does this effect performance?
- Is this topology optimisation approach for SLM practically applicable for the industry?
- Does the simulated performance of topology optimised manifolds approximate reality according to test set-ups?

1.2. THESIS STRUCTURE

The general structure of this thesis is displayed in figure 1.1. The research questions are used to outline the literature survey which serves as the foundation of this thesis. After the literature survey, the main research is distributed in four chapters that will sequentially walk through the topology optimisation approach. Finally, the main research is practically implemented to prove the working principles of this approach.

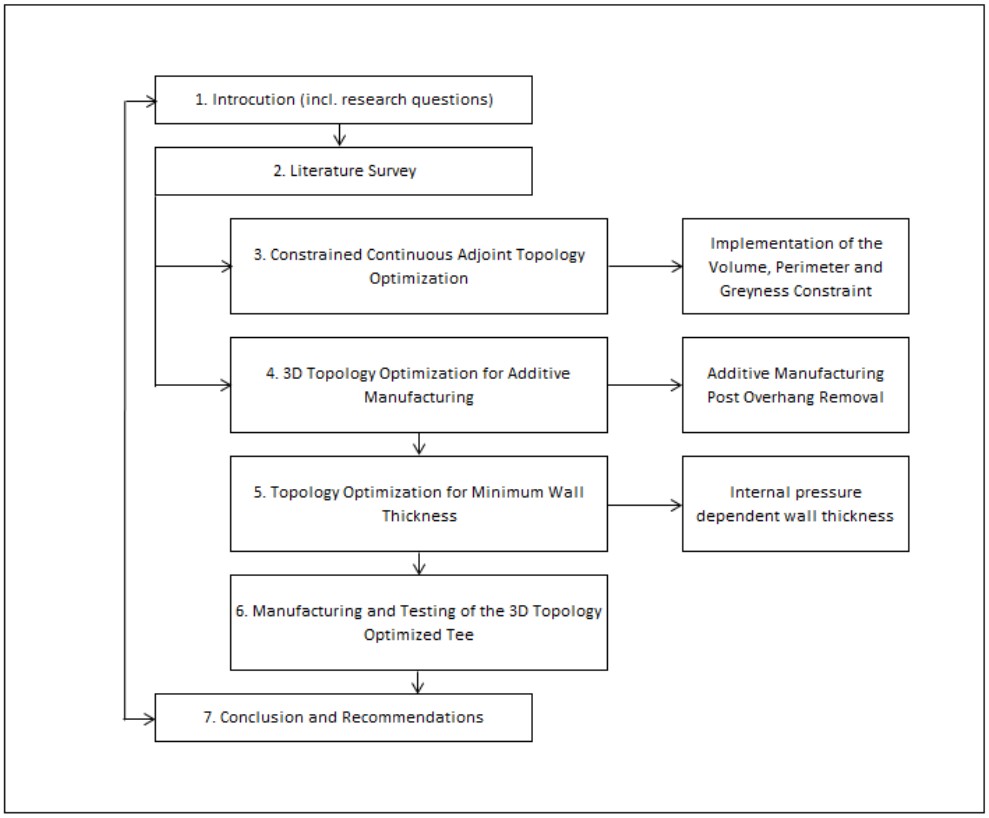


Figure 1.1: Structure of this thesis

2

LITERATURE SURVEY

For this thesis project different topics are surveyed that serves as the foundation for this research. The most notable topics necessary for this project are Additive Manufacturing (AM), in particular, Selective Laser Melting (SLM), which is currently the state of the art technology to print metal parts. Furthermore, optimisation literature is reviewed to create an overview the fast moving research field in both structural and fluid flow optimisation. Finally, we conclude with a review of fluid mechanics which is essential for the fluid flow topology optimisation.

2.1. ADDITIVE MANUFACTURING

Conventional manufacturing techniques (milling, lathing, moulding, etc.) have been used extensively in almost all industries. However, in the 1980s, a new manufacturing method was introduced, Additive Manufacturing (AM), or more popularly 3D Printing. In 2009 patents for fuse deposition modelling (FDM) and 2014 key patents for Laser Sintering methods expired [65]. This opened the doors for the current thriving Additive Manufacturing sector. The fundamental difference from the conventional techniques is that AM adds material where necessary, instead of removing abundant material. This difference allows the designer to develop new geometries with internal cavities or manifolds that were impossible to manufacture before. With AM, the material can be used more efficiently by only adding material where it is necessary. This advantage can, in some cases, result in weight reduction with equal or increased performance.

Over time, multiple AM methods were invented. The method of interest for this thesis research is Selective Laser Melting (SLM) as it can be used for printing metals [12][65][13][21]. The base material for this process is a metal powder such as Stainless Steel, Aluminium and Titanium alloys. It should be noted that SLM is variation from Selective Laser Sintering (SLS). The difference is because SLM fully melts the metal powder whereas SLS just fuses it together on a molecular level. The former requires a much higher energy density to fuse the material entirely. This improvement allows SLM to produces higher quality parts with more homogeneous material properties [13].

Once a three-dimensional model of a part is created it needs to be sliced in layers of a few microns thick, which results in two-dimensional geometries that the SLM printer can read. The printing process then happens in three simple steps:

I Deposit a layer of the metallurgic powder of a 50-300 microns thick

II Melt the current layer with the use of a laser

III Lower the powder bed platform and repeat

Therefore this method is essentially a 2D process that builds layer upon layer. The process is visualised in figure 2.1

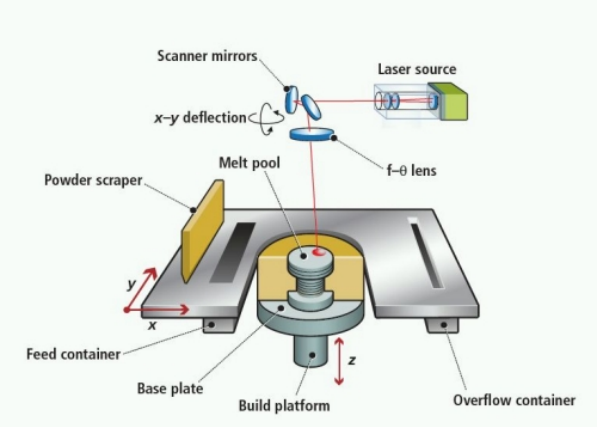


Figure 2.1: Selective laser melting process (Source: popular3dprinters.com)

In figure 2.2 the process is shown in greater detail. The laser melts the powder while it moves in the scanning direction. It shows that there is a colder solidified metal area and the hot melting area. This difference in temperature introduces local stress in the product, which is an actively researched topic. Large solid parts are more prone to the internal stresses and may break during the printing process. A common post process step to remove stress is to put the printed parts in an oven and heat it to a material-specific temperature.

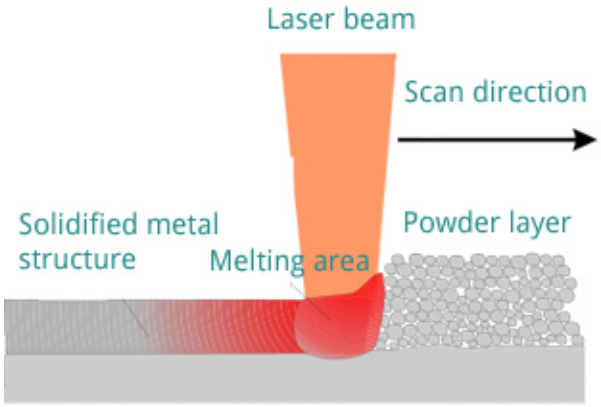


Figure 2.2: Selective laser melting detail (Source: Meiners, 2011)

Although AM technology is improving rapidly, there are still limitations to the manufacturing process. In the following sections, the most notable limitations are discussed that are critical to this project.

2.1.1. OVERHANG CONSTRAINTS

In literature [74][13][33] it is shown that for SLM, the maximum allowed overhang is 45° with respect to the base plate. Below this angle, the accuracy of 1° couldn't be achieved because the structure would start to collapse under its weight. Furthermore, the surface roughness is increased significantly [**?**]. Similar boundary values are used internally at the Dutch Aerospace Laboratory. However, the overhang angle can depend on the printer and material parameters. An experiment to investigate the maximum overhang was repeated for angles between 90° and 45° as is shown in figure 2.3. Thomas showed that below 45° the results were not reproducible and thus setting the design limit at this angle.

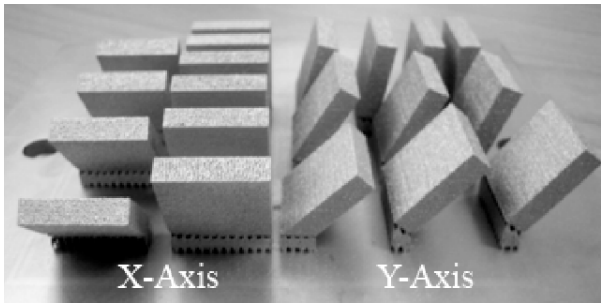


Figure 2.3: Overhang in two different orientations (Source: Thomas [74])

2.1.2. CROSS SECTIONS

Thomas [74], Snyder [71] and Kempen [37] also showed that round holes cannot be manufactured perpendicular to the build plate with the desired accuracy, which is a result of the overhang constraint. Cross sections as small as 1 mm result in sagging material and render them potentially unusable, as shown in figure 2.4. In general, Thomas found that holes smaller than 7 mm can be built with an average of 0.5 mm of sag at the top of the hole. However, larger holes show excessive distortion which may render manifolds useless. In many industries, it is common to have diameters larger than 7 mm. Hence it is important to think of new cross sections that are printable with a constant precision and a better surface roughness, which may result in better flow characteristics.

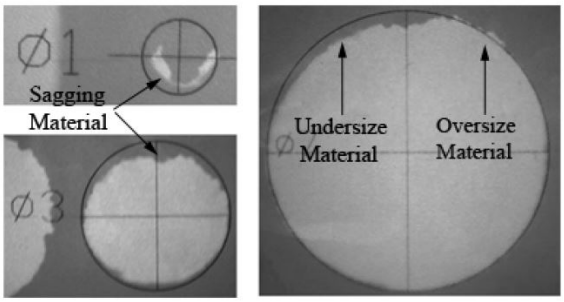


Figure 2.4: Circular cross sections (Sourec: Thomas [74])

Cross sections created with SLM can be constructed of any arbitrary shape in contrast to drilling holes in the solid material. A promising cross section, extensively tested by Thomas, is a round hole with a 45° peak on top as shown in figure 2.5 which ensures the self-supporting property of this cross sections. Experiments showed that increasing the radius of the hole, the more accurate it became, making it a promising substitute for the circular hole. In a different paper by Pakkann [54] a study was performed on the surface roughness of the internal walls. However, no tests were conducted on the actual fluid behaviour with such a cross section. As it is self-supporting, there is no limit on the size of the possible diameters, which make it useful for many applications.

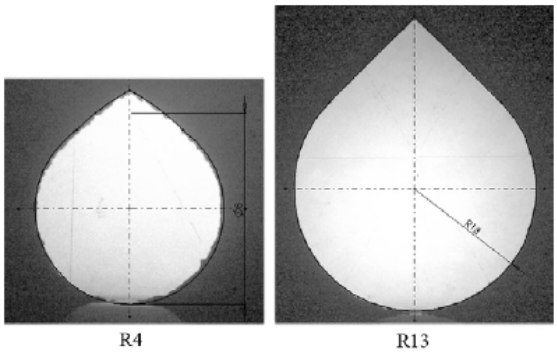


Figure 2.5: New cross section with 45° peak (Sourec: Thomas [74])

2.1.3. SURFACE ROUGHNESS

Metal AM has the disadvantage compared to conventional drilling that the resulting manifolds have a larger surface roughness. The first important observation is that the surface roughness and maximum overhang is somewhat dependent on the chosen material [6] and thus varies per case. It was shown that titanium has a better surface roughness but poorer performance with respect to overhang as compared to aluminium. Secondly, other parameters like print orientation, particle size, particle distribution and other process parameters may influence the surface roughness. These parameters are mentioned for completeness but are outside the scope of this thesis research.

For the application of hydraulic parts, it is important to know what parameters influence this roughness. Because rough internal surfaces increase the turbulent drag which results in a larger pressure drop. Pakkanen [54] investigated the surface roughness of different hydraulic manifold components (Tube, Tee, Reduction and 45° angle) with increasing angles with respect to the build plate of the printer with two different alloys, aluminium (AlSi10Mg) and titanium (Ti6Al4V).

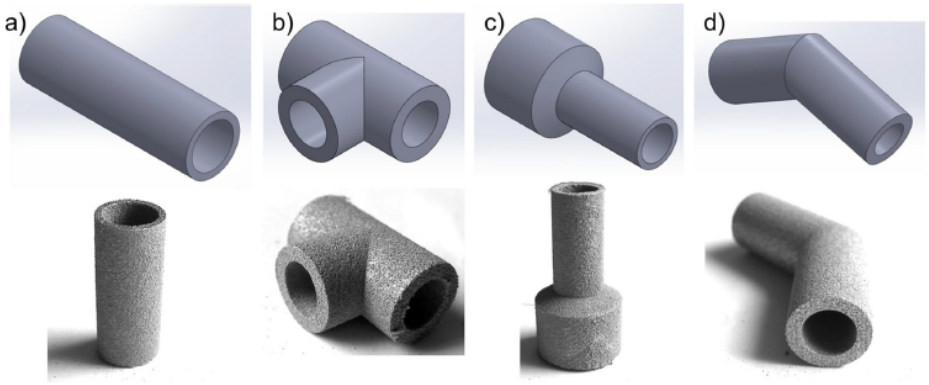


Figure 2.6: Benchmark components for surface roughness (Source: Pakkanen [54])

A pipe or manifold with a circular cross-section that is manufactured perpendicular to the build plate will have 'sagging' behaviour at the top of the cross section [37][74][33], which increases the surface roughness. Pakannen [54] showed that the surface roughness increases as the build orientation become more perpendicular to the build plate, which is a similar conclusion to Thomas.

REDUCING SURFACE ROUGHNESS WITH POST PROCESSING

It is imperative that the surface roughness is as smooth as possible to increase the efficiency of the hydraulic part. However, it is difficult to reach internal cavities and thus not easy to improve the surface roughness with post processing. Several methods are available that can reduce the surface roughness as a post-processing step. In this section, four potential methods are discussed.

1. **Shot peening**
   Shot peening [31][75] is a cold working process that can be used to improve the surface roughness of an object. By impacting the surface with shot particles with high force, one can create actual plastic deformation of the surface. It is similar to sandblasting except that it operates in the plastic regime instead of abrasion. Besides changing the surface roughness, one introduces new surface stresses to the part allowing for some hardening effect. Unfortunately, this method is impractical to use for internal cavities and is only be of limited use.

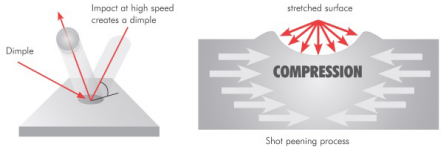


Figure 2.7: Shot peening principle (Source: cwst.co.uk)

2. **Electrolytic Polishing**
   An important requirement for electrolytic polishing [44] is that the objects are conductive. It is known to work well for softer materials like stainless steel and aluminium alloys. Essentially this method dissolves the higher surface energy sites which are associated with rough surfaces by smoothing the peaks. At a certain voltage, the passivation layer becomes stable allowing for the dissolution of the metal. The effect is largest for higher surface regions (peaks) which will dissolve. Hence the surface is smoothed.

   A downside of this method is that the cathode, which is necessary for this process, is quite large. This means that the process can only be used for large internal cavities in which the cathode fits. In figure 2.8 the necessary components are shown; (1) shows the electrolyte fluid which is specifically chosen for a certain material. (2) shows the cathode and (3) is the manufactured part that serves as the anode. (4) Are the loose particles that are being dissolved by the process. At (5) and (6) the initial and the resulting smoothed surface is shown.

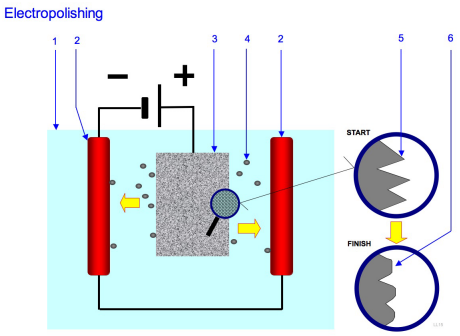
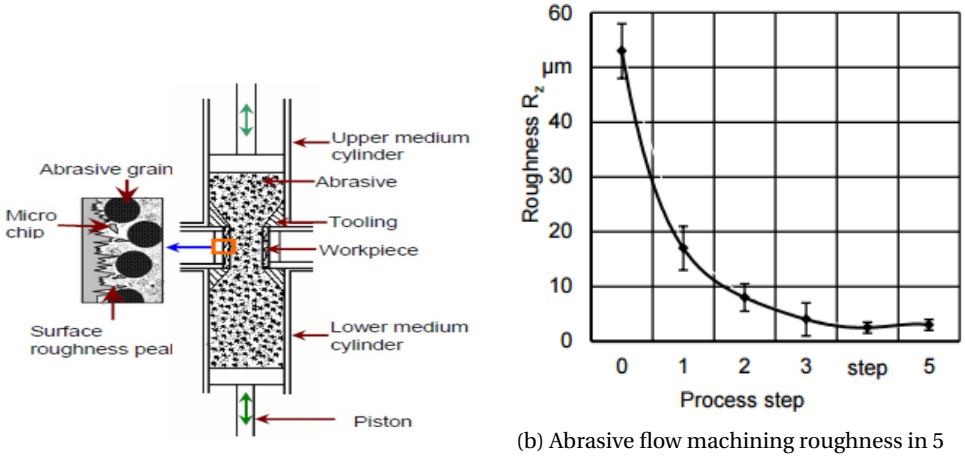


Figure 2.8: Electrolytic polishing principle (Source: wikipedia.org)

3. **Abrasive flow machining (AFM)**
   This method [20] is a process to obtain a smoother surface for internal cavities. Its active substance comprises out of a visco-elastic polymer and abrasive particles in the desired ratio. By extruding the abrasive substance through the pipe or manifold under high pressure, it will smooth the surface up to the desired roughness. After multiple cycles the surface roughness can drop from $R_z = 55\mu m$ to below $10\mu m$ as us shown in figure 2.9.



(a) Abrasive flow machining principle

(b) Abrasive flow machining roughness in 5 cycles

Figure 2.9: Abrasive flow machining (Source: Sankar [68], 2014)

4. **Chemical Polishing**
   Another method to improve surface roughness is chemical polishing. This method is also suitable to reach internal cavities as it works with an etching reagent [61]. It is often used in the micro fabrication processes [34] and commonly called etching. The etching reagent can be pumped through a part to get a homogeneous polishing effect.

   Unfortunately, this method will be actively removing everywhere and thus not specifically smooth. Pyka [61] does show that an increase of active etching ingredients and increasing the treatment time will have a more significant effect on the surface roughness. Hence it can be used for smoothing, but may not yield the best smoothing results.

2.2. TOPOLOGY OPTIMISATION

Different optimisation methods [7] are available to generate new optimised designs such as sizing, shape and topology optimisation. Sizing and shape optimisation methods are limited because a parametric design is necessary. These methods are also limited as they can not introduce new or remove old boundaries. Topology optimisation [9], on the contrary, allows the optimisation algorithm to use the entire design domain. It can close and open new boundaries and therefore introducing more freedom to finding the optimal solution. This method has proved to be very effective regarding generating an initial, non-conventional design. Topology optimisation, first introduced by Bendsøe & Kikuchi (1988), can be summarised as follows:

"Topology optimisation of solid structures involves the determination of features such as the number and location and shape of holes and the connectivity of the domain."
- Bendsøe & Sigmund

Since the introduction of topology optimisation for structural mechanics, it has been expanded for different objective functions, constraints and load cases. Moreover, it was introduced in the multiphysics domain such as fluid flow [10], thermal-fluid interaction, aero-elastics and acoustic-structure problems [16].

In general, topology optimisation is carried out in three steps. The first step is to create a design domain and assign boundary conditions and loads. The design domain needs to be discretized in a mesh for the Finite Element or Finite Volume Method. Secondly, the design evaluation takes place by calculating the governing state equations; Stiffness equation for structural and the Navier-Stokes equations for fluid flow optimisation. The results of the design evaluation are used to calculate the sensitivity of each element and decide if they contribute in minimising the objective (i.e. compliance or power dissipation). In step three the sensitivities of the objective with respect to the design variable is used in a search algorithm like steepest descent. These three steps are being repeated until a certain convergence criterion is met. The topology optimisation loop can be seen in figure 2.10.

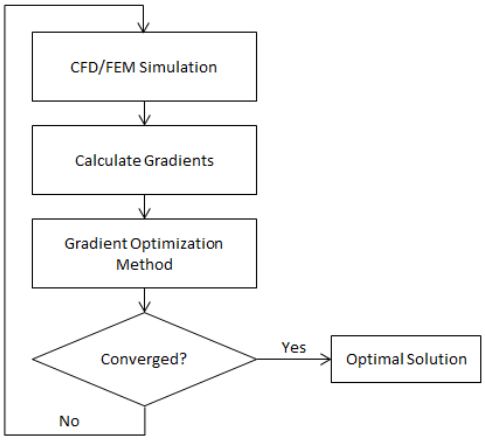


Figure 2.10: General topology optimisation loop

The topology optimisation of fluid flow works similar to the well-known density based structural topology optimisation. However, instead of the governing stiffness equation, the Navier-Stokes (NS) equations are used. In structural topology optimisation, the design variable is usually the density of each element. In the case of fluid flow optimisation, an artificial porosity term α is introduced that takes a similar role to the density variable. The differences between the two optimisation methods are summarised in table 2.1.

	Structural Top Opt	**Fluid Flow Top Opt**
Common numerical method [1]	Finite Element Method	Finite Volume Method
Governing equations	Stiffness Equation	Navier-Stokes Equations
Design variable	Density ρ	Porosity α
Common objective	Compliance	Power dissipation

Table 2.1: Difference between structural and fluid flow topology optimisation

[1] Fluid flow optimisation can also use the finite element method, but the finite volume method is more commonly used for computational fluid dynamics.

2.2.1. STRUCTURAL OPTIMISATION

Structural topology optimisation has a long history in academics tracing back to the 1980s [9], and the most common objective is reducing the compliance, that is increasing stiffness. The most widely used method for structural topology optimisation is called the density-based method. A popular density based method is called the Solid Isotropic Material with Penalization (SIMP) method. The method works with a discretized domain, consisting of small elements. Each element has a density value which is the common design variable in structural optimisation. The algorithm will decide, based on the objective function and constraints, if an element needs to be void or contain material. In general, the minimization problem takes the form as shown in equation 2.1:

$$min: \quad f(\rho, U) \tag{2.1a}$$
$$subject\ to: \quad K(\rho)U = F(\rho) \tag{2.1b}$$
$$g_i(\rho, U) \leq 0 \tag{2.1c}$$
$$0 \leq \rho \leq 1 \tag{2.1d}$$

In equation 2.1, $f(\rho, U)$ is the objective function which is often compliance. This objective function is subject to the governing state equation and constraints. The state equation is in the case of structural optimisation the stiffness equation, where K is the stiffness matrix, U is the displacement vector and F the force vector. It is also possible to implement (in)equality constraints g_i such as a volume constraint. Lastly, the density of an element can take the value in between 0 and 1, where 0 is void, and 1 is full material. It is evident that intermediate values do not make physical sense and should be avoided with a black and white filter, forcing each element to take a discrete value.

For a summary of the work done in topology optimisation, the reader is referred to[16] where the Deaton reported on topology optimisation progress between 2010-2012 which is a great starting point for investigating different methods.

2.2.2. FLUID FLOW OPTIMISATION

It is only since 2003 that topology optimisation for fluid flow started to get traction due to the paper by Borrvall and Petersson [10]. Their idea assumes the design domain to be fully fluid (porous medium) at $t = 0$. Then a local criterion is applied to verify if each element is productive or counter-productive regarding the objective function. This local criterion is typically realised via finite element porosity with the use of Darcy's law. Each element starts with a porosity value of $\alpha_i = 0$ i.e. fully fluid. As an element becomes counter productive, its porosity will be updated until $\alpha_i = 1$, which acts as a solid.

Their novel idea assumed Stokes flow (creeping flow of Newtonian fluids) and thus only very low Reynolds numbers, that is, $Re << 1$ are taken into account and omitting the inertia effect of the fluid. This assumption is only realistic for very viscous fluids like oil or microfluidic applications. However, since Borvall and Petersson introduced, their idea generated a lot of interest from the academic and industrial communities. In 2005 Gersborg-Hansen [25] extended their work to moderate Reynolds numbers and included inertia effects. In 2007 Othmer et al. [52] extended this method with the Navier-Stokes equations in the open source C++ library OpenFOAM® and introduced the topological sensitivities for these problems [51]. Many more developments are currently going on in terms of unsteady flow [39] [38] [40], level-set methods [18] [79]. There are also developments for AM specific topology Optimisation implementations [60] [11].

In the following section, the continuous adjoint method is explained and derived for Navier-Stokes flow, which is chosen for this thesis due to its current implementation in OpenFOAM.

2.3. CONTINUOUS ADJOINT METHOD FOR GRADIENT-BASED OPTIMISATION

The continuous adjoint method is an efficient method to calculate the sensitivities (gradients) of the objective w.r.t. the design variable [3] [26] [35]. This is as a result of the reduced computational effort. Instead of n+1 calculations with the finite differences method, the adjoint method only requires two calculations, also called primal and adjoint equations. When n is large, it is evident that the adjoint method is very effective, which is often the case for CFD simulations that require fine meshes.

The adjoint method comes in various forms such as continuous, discrete or hybrid. The difference between the discrete and continuous method is regarding the order. The starting point for both approaches is a partial differential equation (PDE), but the discrete approach first discretizes these equations and then calculates the adjoint operation. For the continuous method, this is the other way around. For this research the continuous adjoint method is taken into account because it is efficient in run and memory time but most importantly it is implemented in OpenFOAM® [76] [52] [48].

The continuous adjoint topology Optimisation problem takes a similar form as the structural problem:

$$min \quad J = J(\alpha, \mathbf{v}, p) \quad s.t. \quad R(\alpha, \mathbf{v}, p) = 0. \tag{2.2}$$

Where **v** and p describe the velocity and pressure fields, respectively. α is the design variable representing the porosity of each element. Furthermore, the steady-state, incompressible Navier-Stokes equations as shown in equation 2.3a and 2.3b are the state or governing equations. In equation 2.3a the term D(v) is the strain tensor which is $D(\mathbf{v}) = \frac{1}{2}(\nabla\mathbf{v} + (\nabla\mathbf{v})^T)$ and ν is the kinematic viscosity consisting of the molecular and turbulent viscosity such that turbulence can be modeled which will be explained in the section 2.5.4.

$$(R_1, R_2, R_3)^T = (\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p - \nabla\cdot(2\nu D(\mathbf{v})) + \alpha\mathbf{v} \tag{2.3a}$$

$$R_4 = -\nabla\cdot\mathbf{v} \tag{2.3b}$$

It is important to note the extra term $\alpha\mathbf{v}$ which is the extra Darcy term that is the basis of the topology optimisation method. The behaviour of the Darcy term can be explained with two cases. If α is zero, the Darcy term is zero, and thus the standard Navier-Stokes equations are solved, that is, the elements will act as a fluid. On the contrary, if α is large, the Navier-Stokes equation can be neglected, and the elements shall serve as a solid. The porosity variable can range from zero to α_{max}. The latter is chosen such that the velocity values are close zero in the 'solid' region. A rule of thumb is generally when the Darcy number is smaller or equal to $Da = 10^{-5}$ [55]. The definition of the Darcy number is shown in equation 2.4.

$$Da = \frac{\nu}{\alpha_{max} l^2} \tag{2.4}$$

As a result of the governing equation, this problem is a constrained problem. Thus the problem is reformulated with the Lagrange function to arrive at the augmented unconstrained problem.

$$L = J + \int_\Omega (\mathbf{u}, q)\mathbf{R}d\Omega \tag{2.5}$$

In equation 2.5 two Lagrangian multipliers are introduced which are the adjoint velocity **u** and adjoint pressure q. By using variation techniques, the adjoint equations and boundary conditions can be derived [51]. After a long derivation, the adjoint Navier-Stokes equations found as shown in equation 2.6b. As the objective functions are typically described at the in and outlets, the adjoint equations are decoupled from the objective function.

$$-2D(\mathbf{u})\mathbf{v} = -\nabla q + \nabla\cdot(2\nu D(\mathbf{u})) - \alpha\mathbf{u} - \frac{\delta J_\Omega}{\delta \mathbf{v}} \tag{2.6a}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2.6b}$$

For the topology optimisation method, it is necessary to find the sensitivities of the objective function with respect to α such that a new search direction towards the optimal solution is found. These sensitivities could by found by a direct method, meaning an equation for each variable must be solved, which is a very time-consuming process in the case of CFD which works typically with millions of design variables. However, in the case of the adjoint method, it decouples the sensitivity from the design variables, resulting in a much more efficient computation problem. Instead of solving for millions of variables, the solver only has to be called twice, for the primal and adjoint system.

The objective function does not depend on α, hence the sensitivity of the augmented objective with respect to α is the dot product of the adjoint and primal velocity times the volume of an element, as shown in equation 2.8. For the full derivation, the reader is referred to Appendix A.

$$\frac{\partial L}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \int_\Omega (\mathbf{u}, q) \frac{\partial \mathbf{R}}{\partial \alpha_i} d\Omega \tag{2.7}$$

$$\frac{\partial L}{\partial \alpha_i} = \mathbf{u_i} \cdot \mathbf{v_i} V_i \tag{2.8}$$

2.3.1. OBJECTIVE FUNCTION

The objective function of interest for this research is the 'Minimum Power Dissipation' given by equation 2.9 which is measured as the difference in power at the in and outlet. The fluid power is given by the pressure times the volume flow:

$$P = \Delta p \cdot Q = \Delta p \cdot v \cdot A. \tag{2.9}$$

A unit check shows that pressure (J/m^3) multiplied with the volume flow (m^3/s) results in (J/s) which is power (W). This equation can be rewritten in integral form which is the implemented objective function:

$$J_{power} = -\int_{\partial\Omega_{in}} \left(p + \frac{1}{2} v_k^2\right) v_i n_i d\partial\Omega_{in} - \int_{\partial\Omega_{out}} \left(p + \frac{1}{2} v_k^2\right) v_i n_i d\partial\Omega_{out}. \tag{2.10}$$

After a long derivation (see appendix A) the objective function only appears in the adjoint boundary conditions. Thus the implemented solver can be used for many objective functions, as long as the boundary conditions are adjusted.

A different objective function that could be of interest for manifolds is the flow uniformity at an outlet. This objective function was introduced by Othmer [51]. The objective is formulated as shown by equation 2.11 where $\mathbf{v}^d$ is the user defined velocity at the outlet and c is a constant for unit consistency.

$$J_{uniform} = \int_{\partial\Omega_{out}} \frac{c}{2} (\mathbf{v} - \mathbf{v}^d)^2 d\partial\Omega \tag{2.11}$$

2.3.2. 'ONE-SHOT' METHOD

Since the state equations and design variables are independent due to the nature of the continuous adjoint method, they can be solved simultaneously for each iteration. With this approach, the design variable (porosity) can be updated after each CFD iteration with non-converged velocity and pressure fields. That is, the primal and adjoint equations are not fully converged before the next optimisation step (steepest descent) is taken. This is a result of Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm, which is the foundation of the OpenFOAM solver called 'simpleFoam'.

SIMPLE ALGORITHM

The SIMPLE algorithm [64] [43] is widely used in the field of CFD for solving the Navier-Stokes equations. It is essentially a guess-and-correct procedure [77] to calculate the pressure, which is unknown. The SIMPLE algorithm is initialized by guessing the pressure and velocity fields, which are used to solve the momentum equations. Then four correction terms are introduced, three for the velocity components and one for the pressure. The correct pressure is then found by adding the guessed pressure and correction term. By substituting the correct pressure in the momentum equations will yield the correct velocity field. This process is continued until a user defined convergence criterion is met. The algorithm loop is visualized in figure 2.11.

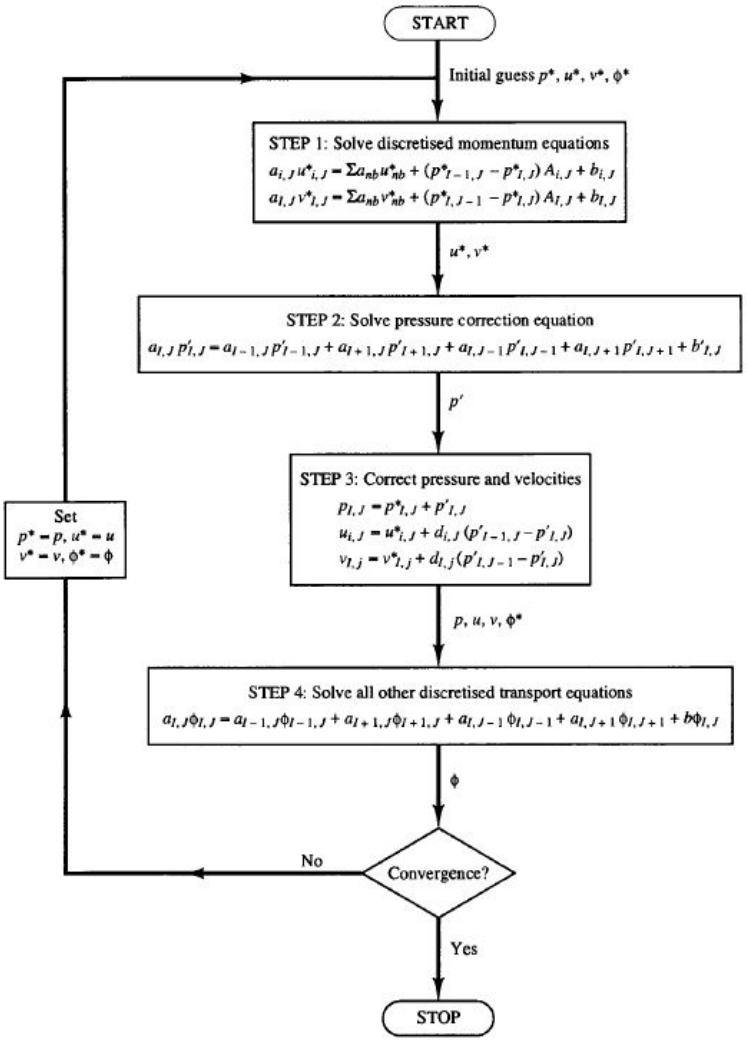


Figure 2.11: The SIMPLE algorithm (Source: Versteeg 1995 [77])

2.4. TOPOLOGY OPTIMISATION SOFTWARE

There is currently a wide variety of commercial packages available for structural topology optimisation which are integrated into FE packages like Abaqus and Comsol. Besides commercial software packages, there have been different open source developments like TopOpt from the Denmark Technical University (DTU) including the well known 99 line code, ToPy a python library and PolyTop which is written in Matlab. Most of the Open Source structural topology optimisation work is presented in 2D but can be extended to 3D if desired.

For fluid topology optimisation, on the other hand, there is not as much software available because the methods are relatively new. However, 3DS recently introduced a commercial fluid optimisation package

for Abaqus TOSCA. The multiphysics package Comsol also comes with some basic fluid topology optimisation. Unfortunately, these packages are not available for this thesis project, so no further research is done on what methods are implemented. To the author's knowledge, there are currently two open source codes available: PolyTop for fluids using their mesh and FE code in `Matlab`, following Borrvall and Petersson's method for Stokes flow. The second option is OpenFOAM [27] [28] which is first and foremost a C++ library. It is a scripted open source Computational Fluid Dynamics (CFD) package that includes a continuous adjoint solver called: `adjointShapeOptimisationFoam`. This solver has been implemented by Othmer et al. [52] and showed the possibilities for fluid topology optimisation. Furthermore, Ulf Nilsson [48] reported on the implementation of different objectives for a course at Chalmers University. In table 2.2 the capabilities of different software packages are shown.

Program	CFD	Shape Opt	Topology Opt	3D	Open Source
Comsol	✓	✓	✓	✓	
Abaqus TOSCA Fluid	✓	✓	✓	✓	
OpenFOAM (C++)	✓	✓	✓	✓	✓
PolyTop Fluid (Matlab)			✓		✓[2]

Table 2.2: Available software for fluid topology Optimisation

2.5. FLUID MECHANICS REVIEW

For this research, optimisation of Navier-Stokes flow in internal manifolds is considered. Therefore, it is essential to review some important aspects of fluid mechanics and computational fluid dynamics [77]. A fluid element is mathematically defined by considering vast numbers of molecules such that molecular interaction can be omitted such that it can be modelled as a continuum. Fluid deforms under the influence of shear forces such that the deformation velocity is a function of the shear forces, whereas the deformation of a solid is a function of the shear force itself [78]. In equation 2.12a the shear stress τ is directly proportional to the resulting strain rate where $\delta\theta$ is shear strain angle. This can also be represented in the velocity gradient which results in 2.12b where μ is the constant viscosity coefficient. In relation to this, friction losses in hydraulic systems are due to the shear stress between the surface of the pipe and the fluid.

$$\tau \propto \frac{\delta\theta}{\delta t} \tag{2.12a}$$

$$\tau = \mu \frac{du}{dy}. \tag{2.12b}$$

The Reynolds number, given in equation 2.13, is the primary (dimensionless) parameter to correlate viscous behaviour and provides information about the flow behaviour. It is defined by the inertial forces (density ρ times velocity v times the hydraulic diameter D) divided by the viscous forces (dynamic viscosity).

$$Re = \frac{Inertial \quad Forces}{Viscous \quad Forces} = \frac{\rho \cdot v \cdot D}{\mu} \tag{2.13}$$

There are three defined regimes for circular pipe flow. In the first regime, the viscous forces are dominant, and the flow is laminar. The second regime is the transition regime between laminar and turbulent flow, which is a regime that is typically avoided by engineers due to its random nature. In the last region when the Reynolds number is higher than 3500, the flow is fully turbulent. It is important to note that these ranges vary with the flow geometry, surface roughness and the fluctuations in the inlet stream.

[2] The code is open source however one needs Matlab for this option.

Table 2.3: Reynolds ranges for a circular pipe

Reynolds	**Flow**	**Dominant Force**
$Re < Re_{crit}$	Laminar	Viscous
$Re \approx Re_{crit}$	Transition	Equally
$Re > Re_{crit}$	Turbulent	Inertial

For low Reynolds numbers, the viscous forces are dominating i.e. high viscosity. The inertial forces will dominate for high Reynolds numbers, which is the case if the density, velocity and hydraulic diameter are large. The Reynolds number, therefore, says something about the flow regimes which can be summarised as shown in 2.3. The flow regimes are often visualised with ink in a straight pipe as is shown in figure 2.12. From top to bottom the laminar flow, transitioning flow and lastly, turbulent flow is shown.

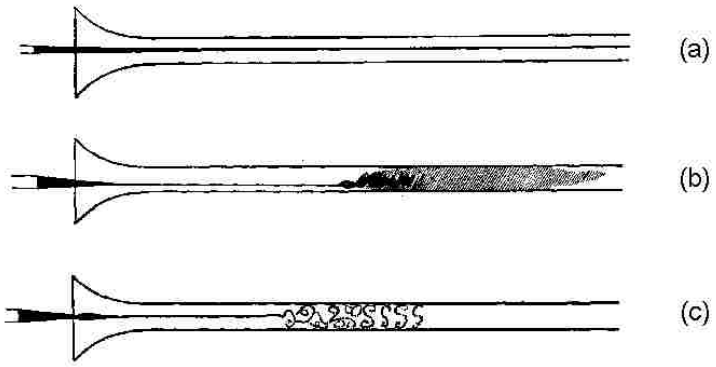


Figure 2.12: Visualisation of different flow regimes (Source: Osborne Reynolds, 1883)

2.5.1. TURBULENCE

Turbulence is a very complex phenomenon that has been studied for centuries, starting with Da Vinci up to Reynolds and many more great scientists [67]. Despite their efforts, we still do not understand in full how or why turbulence occurs. It is not yet possible to accurately predict the behaviour of turbulence, and thus it must be modelled the best we can. A well-known definition of turbulence is given by Richardson [46]:

Big whorls have little whorls,
which feed on their velocity;
And little whorls have lesser whorls,
And so on to viscosity.

This definition appends the idea that the initial kinetic energy is cascaded into smaller scales until it is dissipated and turned into thermal energy. The reader should note that this is only one of the many definitions of turbulence. However, there is currently no definition gives a precise description of turbulent flow.

In 1880, Osborne Reynolds observed for the first time the correlation of flow velocity, pipe diameter, and viscosity by using inked water. He found out that laminar flow can recover from disturbances at low velocities. Moreover, at higher velocities, it would not be able to recover from these disturbances, and the flow would become unstable. This behaviour was explained as he found out by the Reynolds number. When Re is low, the viscous effect is dominating, and this stabilises the flow from perturbations. As Re keeps growing the viscous effect become negligible and small perturbations in the flow can grow out to unstable flows into a turbulent flow. When flow becomes turbulent so-called eddies appear. Physically the initial kinetic energy is converted from larger eddies into smaller eddies until they are dissipated as heat energy on a molecular scale. The flow direction and magnitude of eddies differs from the general flow direction and show velocity fluctuations in the velocity field.

2.5.2. STEADY-STATE INCOMPRESSIBLE NAVIER-STOKES FLOW

This research takes into account steady state incompressible fluids which are modeled by two state equations 2.14. The first equation follows directly from the momentum conservation, which is also known as the Navier-Stokes equation. The second equation follows from mass conservation or also referred to as the continuity equation. Two assumptions are made to arrive at these equations. Steady-state means that the time dependent terms are set to zero and therefore no accelerations are modeled. Assuming that a fluid is incompressible means that the density is independent of the pressure field. The full Navier-Stokes equations have the following form:

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p - \nabla\cdot(\nu\nabla\mathbf{v}) = 0 \tag{2.14a}$$

$$\frac{\partial \rho}{\partial t} + \rho\nabla\cdot\mathbf{v} = 0. \tag{2.14b}$$

After applying the assumptions mentioned above, the steady-state incompressible Navier-Stokes equations are found.

$$(\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p - \nabla\cdot(\nu\nabla\mathbf{v}) = 0 \tag{2.15a}$$

$$\nabla\cdot\mathbf{v} = 0. \tag{2.15b}$$

In the event of a very small Reynolds numbers i.e. $Re << 1$ the flow is called creeping flow. This flow regime is also called Stokes flow another assumption is made. Because the inertial forces can be neglected for creeping

flow, the Navier-Stokes equations turn into the Stokes equations:

$$\nabla p - \nabla \cdot (\nu \nabla \mathbf{v}) = 0 \tag{2.16a}$$
$$\nabla \cdot \mathbf{v} = 0. \tag{2.16b}$$

2.5.3. REYNOLDS AVERAGED NAVIER-STOKES (RANS) EQUATIONS

Unfortunately, it is not possible to solve the NS equations for turbulent flow due to its non-linear and coupled (velocity-pressure) characteristics. A turbulent flow field has velocity fluctuations in three dimensions and has an infinite number of scales (large through small eddies). A solution to these infinite degrees of freedoms is the Reynolds decomposition. This approach decomposes the velocity field $\overline{v}$ in a mean V and fluctuation v' quantity. The idea is visualised in figure 2.13 where u replaces v.

$$\overline{v} = V + v' \tag{2.17}$$

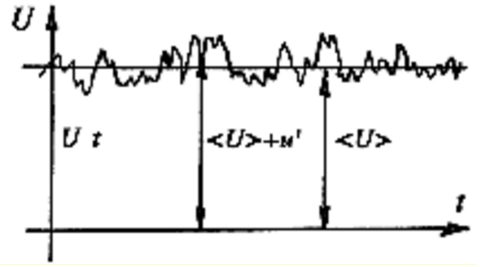


Figure 2.13: Mean velocity U and fluctuating velocity u' (Source: daad.wb.tu-harburg.de)

If we take the mean velocity V and the fluctuation v' and substitute it in the NS equations in 2.14, the Reynolds averaged Navier-Stokes (RANS) equations [46] are found. These new equations will be exact for the mean flow field but not for the turbulent flow field due to the average nature.

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} + \nabla p - \nabla \cdot (\nu \nabla \mathbf{v}) = -\nabla \cdot \mathbf{v}' \otimes \mathbf{v}'. \tag{2.18}$$

As a result of the Reynolds decomposition, a new term called the Reynolds stress tensor appears in this equation:

$$\tau_{ij} = -\left(v_i' v_j'\right). \tag{2.19}$$

Equation 2.19 leaves the equations with a closure problem in the sense that there are more unknowns than equations (6 Reynolds stresses and three turbulent fluxes). For this purpose, different turbulence models are available to provide an approximation for the Reynolds stress tensor. It is possible to derive new PDEs for each term, but this quickly becomes very complicated due to new correlations. Many models are available and listed below in the order of complexity:

- First Order Models
  - Zero Equation Models
  - One-Equation Models
  - Two-Equation Models (k-ϵ)
- Second Order Models
  - Algebraic Stress Models (ASM)
  - Reynolds Stress Models (RSM)

2.5.4. TURBULENCE MODELING: EDDY VISCOSITY MODELS (EVM)

The model of interest is a first order model or also referred to as the Eddy Viscosity Model (EVM). It is advantageous because it will only introduce two new partial differential equations (PDEs) at most. Another option could be the ASM/RSM, but they are second order and are much more complex regarding implementation and computational power. The EVM is based on the Boussinesq assumption (1877) which states that momentum transfer due to turbulent eddies can be modelled by an eddy viscosity. The analogy between laminar and turbulent flow is that the average turbulent flow field is similar to the corresponding laminar flow. In equation 2.20 the rate of strain tensor is presented that includes the eddy viscosity term ν_T.

$$\tau_{ij} = \rho\left(v_i' v_j'\right) = \nu_T\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}k \tag{2.20}$$

The deviator of the Reynolds stress tensor is assumed to be proportional to the mean shear rate, and the proportionality factor is the eddy viscosity ν_T. It should be noted that the this is not a material property such as the kinematic viscosity but merely a model. Substituting this model in the RANS equation gives us the closure we sought because the stress tensor is reduced from six independent components to one scalar field, the eddy viscosity. This eddy viscosity can be expressed as a function of velocity in many ways, and in the next section, the $K-\epsilon$ model will be discussed.

$k-\epsilon$ MODEL: TWO EQUATIONS

The k-epsilon model is one of the most used EVM models in the industry and academics because of its low computational effort and only two extra transport equations. The idea is to express the turbulent viscosity ν_t as a function of the turbulence kinetic energy k and the turbulence dissipation rate ϵ. The turbulent viscosity equation is defined as follows:

$$\nu_t = C_\nu \frac{k^2}{\epsilon}. \tag{2.21}$$

Thus, two new PDEs must be solved for k and ϵ which can be found in Appendix F. The $K-\epsilon$ model should not be used for models with high-pressure gradients or separation effects. Moreover, it also cannot predict anisotropic effects such as streamline curvature and volume forces such as gravity.

FROZEN TURBULENCE

A common assumption is to take the eddy viscosity as constant and therefore the derivatives will be zero, which is called the Frozen Turbulence assumption [50]. This is valid for laminar flow but is only an approximation for turbulent flows of low intensities i.e.

$$u'/U << 1. \tag{2.22}$$

In this equation, U is the mean velocity and u' is the eddy velocity or the fluctuations. As ν_t is assumed constant, it means for the $K-\epsilon$ model that the derivatives of the turbulent viscosity are zero, which makes it easier to implement and reduce the computational cost for the adjoint method. To give an order of magnitude of turbulence intensity different intensity estimations, [8] [66] can be considered. For fully developed pipe flow the intensity can be estimated as follows (as implemented in Ansys):

$$I = 0.16 Re^{-\frac{1}{8}}. \tag{2.23}$$

For a turbulent flow with Reynolds number of 50.000, the intensity is approximately I = 0.04, which can still be considered in the range of frozen turbulence. Since Re = 50.000 is the maximum Reynolds number that is likely to be used during this research, it can be assumed this to be valid.

3

CONSTRAINED CONTINUOUS ADJOINT TOPOLOGY OPTIMISATION FOR NAVIER-STOKES FLOW

In the literature survey the most important topics regarding the continuous adjoint method for Navier-Stokes flow have been discussed. Currently, the implementation in OpenFOAM is unconstrained and therefore of limited practical use. In this chapter, two constraints are added to the solver that are important for different reasons. Firstly the standard volume constraint is implemented that is used to verify our results with present papers [55] [10]. Without this constraint, the solver is less stable and may not converge to a closed geometry, which is quite essential if the final geometry needs to be manufactured. Furthermore, this constraint gives a very intuitive and easy way to tune the size and indirectly the weight of the manifold. The second constraint, a perimeter constraint, is more important for several reasons. The first and foremost reason is that the perimeter of a manifold is directly proportional to its weight. In two dimensions the perimeter will be a length, and in three dimensions it will be an area. By penalising this, the constraint will reduce the length or area. Furthermore it can be used similarly to the volume constraint to find closed geometry solutions, but it is less intuitively implemented.

3.1. VOLUME CONSTRAINT FOR STABILITY AND INDIRECT MASS CONTROL

The current implementation is rather unstable and doesn't converge to clean solutions for low Reynolds numbers as well as high Reynolds numbers. By implementing a volume constraint the solver converges faster and more stable towards a new design. It is also an important parameter to verify the results from OpenFOAM with existing results. For this purpose the proposed volume constraint given in [55] is used and is defined as shown in equation 3.1:

$$c_{vol} = \left[\frac{\int_\Omega (1 - \frac{\alpha}{\alpha_{max}}) d\Omega}{\int_\Omega d\Omega} - V_{target} \right]^2 . \tag{3.1}$$

In this equation α is the design variable, α_{max} is the maximum value alpha is allowed to take and finally V_{target} is the user defined target volume fraction. This constraint has to be added to the existing implementation and will only appear in the sensitivity of the augmented objective function with respect to the design variable α. The Augmented Lagrange Multiplier (ALM) method is used to implement the volume constraint because both the constraint and its derivative are known. The ALM tends to be more precise and has less trouble with ill-conditioning. In general, the constraint will be applied as shown in equation 3.2 where J is the objective function, R are the state equations as presented in equation 2.3a and 2.3b. The variables **u**, q and λ_{vol} are the Lagrange multipliers and w_{vol} is the volume weight factor

$$L = J + \int_{\Omega} (\mathbf{u}, q) R - \lambda_{vol} c_{vol} + w c_{vol}^2 d\Omega. \tag{3.2}$$

Since the volume constraint is only dependent on the α field, it will appear in the sensitivity of the adjoint formulation. That is, the derivative of the augmented Lagrange formulation with respect to α

$$\frac{\partial L}{\partial \alpha} = \frac{\partial J}{\partial \alpha} + \int_{\Omega} v_i u_i - (\lambda_{vol} + 2 w c_{vol}) \frac{\partial c_{vol}}{\partial \alpha} d\Omega. \tag{3.3}$$

The objective function is not dependent of α as seen in equation 2.10 and thus the derivative of the objective on the r.h.s. will be zero. This lead us to the final sensitivity that is implemented in the OpenFOAM solver

$$\frac{\partial L}{\partial \alpha} = \int_{\Omega} v_i u_i d\Omega + \int_{\Omega} (\lambda_{vol} + 2 w c_{vol}) \frac{\partial c_{vol}}{\partial \alpha} d\Omega. \tag{3.4}$$

The last step is to derive the volume constraint and substitute the result back in equation 3.4. The derivative of the volume constraint is shown in equation 3.5.

$$\frac{\partial c_{volume}}{\partial \alpha} = -\frac{2}{\alpha_{max} \int_{\Omega} d\Omega} \left[\frac{\int_{\Omega} (1 - \frac{\alpha}{\alpha_{max}}) d\Omega}{\int_{\Omega} d\Omega} - V_{target} \right] \tag{3.5}$$

The original sensitivity given by the dot product of the primal and adjoint velocity is unique for each element. The volume constraint, on the contrary, has the same value for each element. As long as equation 3.6 holds for a given element; the element will stay open and fluid may flow.

$$u_i v_i > (-\lambda_{vol} + 2 w_{vol} c_{vol}) \frac{\partial c_{vol}}{\partial \alpha} \tag{3.6}$$

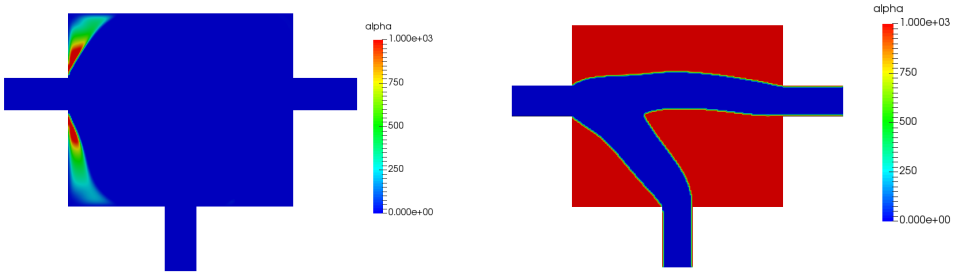
After each iteration, the Lagrange multiplier and weight factor are being updated. Finally, it is important to note that the initialization values and the updated values for the Lagrange multiplier and weight factor are chosen such that the constraint value is in the order magnitude of the original sensitivity. If the value of the volume constraint is too large, the alpha field will close the design field quickly. This approach will not result in an optimal solution since the volume constraint is dominating over the actual flow sensitivity. On the other hand, if the volume constraint is too small, essentially an unconstrained problem is solved. This implies that the parameters need to be tuned for each specific case. A good solution for this is to run the simulation once and find the maximum values of both terms of the sensitivity. These values can be used to initialize the parameters to ensure they are of the same order.

3.1.1. RESULTS WITH VOLUME CONSTRAINT

As discussed before, without the volume constraint the optimisation algorithm does not always converge to a useful design. This behaviour depends mostly on the primal and adjoint velocities because the sensitivity is the dot product of these two quantities. To show the difference between the unconstrained and constrained solver, a simple tee manifold is used as an example. Both a low and high Reynolds number case will be discussed.

In figure 3.1 the unconstrained behaviour for low Reynolds numbers is visible. Without the volume constraint, the algorithm will not converge to a proper design because the full design domain is an optimal solution. This shows that the volume constraint is essential in finding practical low Reynolds number results. Even though the result without the volume constraint is not wrong, it is not a practical design. First an foremost this constraint allows the user to find different designs for different allowable volume percentages. Which is also loosely linked to the weight of a pipe or manifold, and can, therefore, be a simple tool to balance performance versus weight.

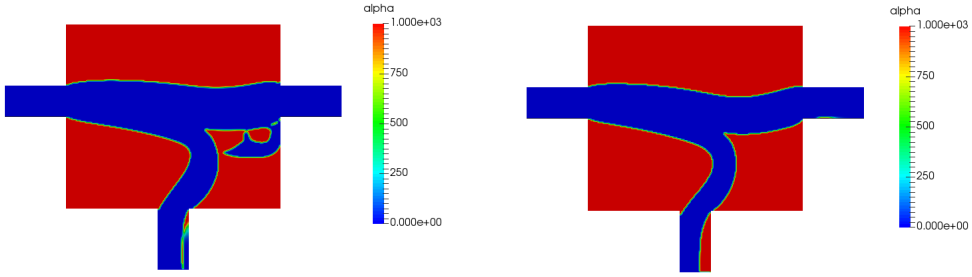
For high Reynolds numbers, the solver without the volume constraint has difficulties to find stable solutions, depending on the problem at hand. In the case of the tee manifold, the solver finds a decent solution. However, there are still some instabilities visible near both outlets, as shown in figure 3.2. These open regions tend



(a) Unconstrained α field (b) Volume Constraint α field

Figure 3.1: Manifold with low Reynolds number (Re = 10)

to grow and result in a design that performs properly but is not feasible regarding manufacturing. In some cases, there are open areas in the nonporous region ($\alpha = \alpha_{max}$). However, since there is no changing flow in such an area, it has no effect on the objective value.



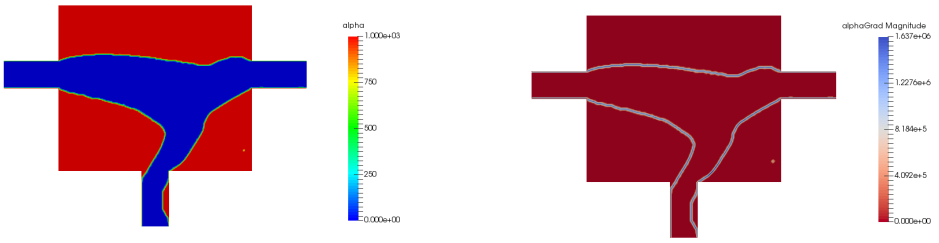
(a) Unconstrained α field (b) Volume Constraint α field

Figure 3.2: Manifold with high Reynolds number (Re = 50000)

3.2. PERIMETER CONSTRAINT FOR DIRECT PROPORTIONAL MASS CONTROL

A volume constraint is a simple tool that gives the user freedom to choose the amount of artificial porous to nonporous (fluid to solid) ratio. It is unfortunately not directly proportional to the mass of a manifold. A perimeter constraint, on the contrary, is directly proportional to the mass of the manifold, if we assume the wall thickness to be constant. By constraining the perimeter i.e. length (2D) or area (3D), the constraint will proportionally reduce the mass of the manifold. Perimeter constraints have been used by Allaire et al. [2] and Qian [63], who both implemented it in a structural topology optimisation code for different purposes.

The design field α goes from zero to α_{max}, and thus the boundary between porous and nonporous material is very sharp. This characteristic of the design field can be used to set up the perimeter constraint. By taking the gradient of the design field, we find a sharp line that represents the perimeter of the manifold in 2D as shown in figure 3.3b.



(a) Design field α (b) Magnitude of the Gradient of α

Figure 3.3: Design field α and the magnitude of the gradient

It should be noted that this gradient field also holds directional information of the boundary which is shown in figure 3.4. The change from porous to nonporous material and vice versa, yields this directional information that is useful for the perimeter constraint.

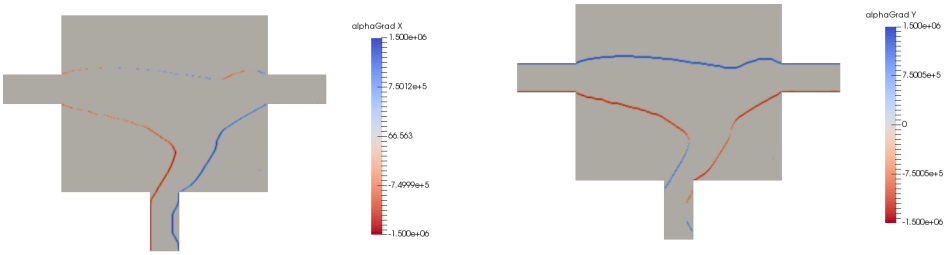


Figure 3.4: Gradient of alpha in the x and y direction

The perimeter can be found by taking the integral of the current design i.e. lines (dx) in 2D or surfaces (ds) in 3D.

$$c_{perim} = \int ds. \tag{3.7}$$

For the implementation of the perimeter constraint, we seek the derivative of the perimeter with respect to the design variable α. Allaire [2] mathematically proves that the derivative of the perimeter in 3D is equal the mean curvature (H) of the perimeter. The mean curvature of the perimeter can be calculated by taking the divergence of the unit normal of the perimeter, which represents the outward flux of the vector field:

$$H = \nabla \cdot \vec{n}. \tag{3.8}$$

The unit normal is based on the gradient of the design field, as is shown in equation 3.9, which is the derivative of the perimeter constraint. ϵ is chosen very small to avoid numerical errors if the magnitude of the gradient of α is zero. Finally, we multiply the curvature with the magnitude of the gradient of α such that this constraint is only active on the perimeter of the manifold. This generates the field in figure 3.5 which represents the perimeter constraint

$$\frac{\partial c_{perim}}{\partial \alpha} = \nabla \cdot \left(\frac{\nabla \alpha}{|\nabla \alpha| + \epsilon} \right) \cdot |\nabla \alpha|. \tag{3.9}$$

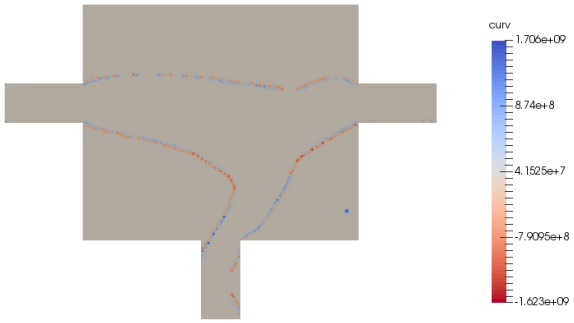


Figure 3.5: Curvature of the design field $alpha$

The perimeter constraint can be satisfied by using intermediate α values i.e. a grey design. The results are mathematically correct, but the resulting design is unusable, as intermediate values have no physical meaning. To avoid the intermediate behaviour, a greyness constraint is implemented that forces each element to take a value of 0 or α_{max} and will be introduced in the next section.

3.2.1. GREYNESS CONSTRAINT

As stated in the previous section a greyness constraint is necessary to force the perimeter constraint to find a 'black & white' design. This means that each element can take a value of either zero or α_{max}, but nothing intermediate. Without this constraint, the perimeter constraint will result in unusable design as is shown in figure 3.6.

(a) Design field if there is a no or a very low greyness constraint

(b) Design field if there is a small valued greyness constraint

Figure 3.6: Behaviour of the perimeter constraint with no or a small valued greyness constraint

Each element can be forced to take a zero value or a maximum value by formulating the following equation:

$$c_{grey} = \frac{\alpha}{\alpha_{max}}\left(1 - \frac{\alpha}{\alpha_{max}}\right). \tag{3.10}$$

If we assume $\alpha_{max} = 10$ it can be seen that c_{grey} is zero when either $\alpha = 0$ or $\alpha = 10$. In these cases the elements are already 'black & white'. However as the value of α goes further away from either extreme, the value of the constraint becomes larger and thus more active. In this manner the constrain tends to drive each element to a black or a white solution. The derivative that we need for our sensitivity implementation is found as follows:

$$\frac{\partial c_{grey}}{\partial \alpha} = \frac{\left(1 - 2\frac{\alpha}{\alpha_{max}}\right)}{\alpha_{max}}. \tag{3.11}$$

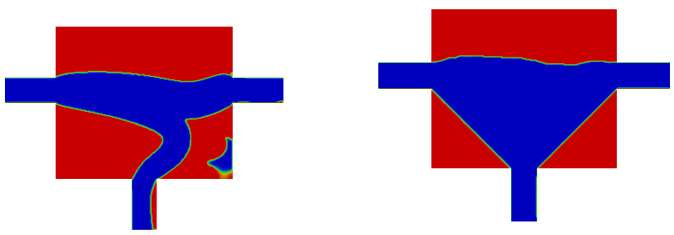
The greyness constraint has to be implemented together with the perimeter constraint. However, choosing values for the Lagrange multipliers is a very delicate task to ensure good convergence. In the next sections, we will discuss the behaviour of the combination of these two constraints.

3.2.2. RESULTS WITH THE PERIMETER AND GREYNESS CONSTRAINT

Both the perimeter and greyness constraint have been implemented with a simple weight factor and not with the Augmented Lagrange Method (ALM). The advantage of a simple weight factor is because only the derivative information is necessary. However, this comes at a cost because the ALM is more precise and has less trouble with ill-conditioning [49].

The perimeter constraint is implemented to directly control the mass of a pipe or manifold by constraining its perimeter. If we set the perimeter constraint rather aggressively, we expect to find straight lines (in 2D) between in and outlets, because it is the shortest path i.e. the smallest perimeter. This is a result of the divergence term, which pushes the boundaries in or outward, such that the curvature is reduced.

In figure 3.7 the behaviour of the perimeter constraint for a 2D manifold is shown. As expected, the final geometry consists of straight lines, which are shorter than the more organic perimeter of the initial design at the first iteration. It is evident that in the 2D manifold case, the resulting geometry is a terrible design, but it proves the working principle of the constraint.

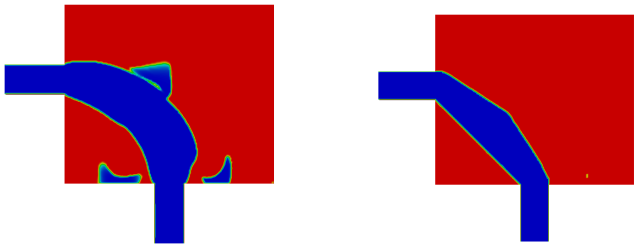


(a) Unconstrained α field before perimeter and greyness constraint become active

(b) Final iteration with perimeter and greyness constraint

Figure 3.7: Manifold result of the perimeter and greyness constraint

The same behaviour can is found for a 2D elbow, and it also gives the result we expect. Starting from a curved elbow, which is expected from Navier-Stokes flow we again find a very straight solution. In figure 3.7 we see the straight elbow with a small bend in the top perimeter. If we would simulate further the final design should be a straight pipe connecting the in and outlet. However, this may take a very long time as the curvature is almost zero in the small bend.



(a) Unconstrained α field before perimeter and greyness constraint become active

(b) Iteration 1500 with perimeter and greyness constraint

Figure 3.8: Elbow result of the perimeter and greyness constraint

3.2.3. SENSITIVITY STUDY FOR THE PERIMETER AND GREYNESS CONSTRAINT

Choosing the right parameters for the constraints are essential to finding a meaningful design. The following sensitivities and parameters in table 3.1 are identified as important

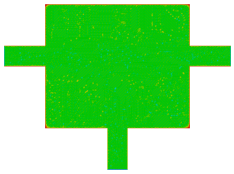
Table 3.1: My caption

Parameter	**Symbol**
Unconstrained Sensitivity	$v_i \cdot u_i$
Perimeter constraint	$\frac{\partial c_{perim}}{\partial \alpha}$
Perimeter weight factor	λ_{perim}
Greyness constraint	$\frac{\partial c_{grey}}{\partial \alpha}$
Greyness weight factor	λ_{grey}

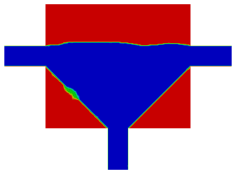
The first thing that must be taken into account is the order of magnitude of each parameter. The three terms i.e. old sensitivity, perimeter constraint, and greyness constraint must be in the same order of magnitude. If the value of the perimeter constraint is too large, the alpha field will be fully filled, and if it is too small, there will hardly be any effect. Secondly, the ratio between the perimeter and greyness constraint have a big effect on the result.

A sensitivity study shows how the constraint reacts to small changes in the weight factors. It can be seen in figure 3.10 that minor changes of the weight factor λ_{perim} results in significant changes to the design that quickly makes it unusable.

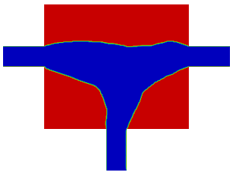
First, the behaviour of the design is considered as the greyness weight factor λ_{grey} is increased. The first design in figure 3.9a shows that the value of λ_{grey} is too small because the design is still able to find an intermediate design. Note that in this figure the greyness is displayed in green, which means that the algorithm tries to use prohibited intermediate values. As the greyness weight factor is increased the design changes. In figure 3.9b it can be seen that the inlet and outlets are connected by almost straight lines, which in 2D is the smallest perimeter. In theory, this design is the most optimised design regarding mass as a result of the perimeter constraint. However, this design is not of any practical use. As λ_{grey} is increased the design in figure 3.10a is found. When λ_{grey} gets too large as seen in the last two figures in figure 3.10 the design will not be able to change anymore, because the greyness constraint value becomes too large.



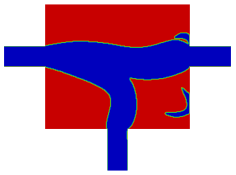
(a) Lagrange multipliers: $\lambda_{perim} = 4e-10$ and $\lambda_{grey} = 1e2$. Objective Function: $J = 0.004175$



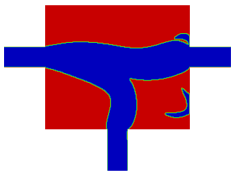
(b) Lagrange multipliers: $\lambda_{perim} = 4e-10$ and $\lambda_{grey} = 5e2$. Objective Function: $J = 0.0001752$



(c) Lagrange multipliers: $\lambda_{perim} = 4e-10$ and $\lambda_{grey} = 1e3$. Objective Function: $J = 0.0001504$



(d) Lagrange multipliers: $\lambda_{perim} = 4e-10$ and $\lambda_{grey} = 5e3$. Objective Function: $J = 0.0001222$

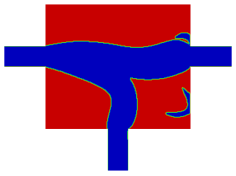


(e) Lagrange multipliers: $\lambda_{perim} = 4e-10$ and $\lambda_{grey} = 1e4$. Objective Function: $J = 0.0001222$

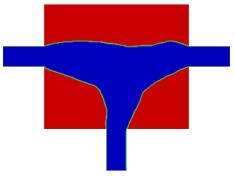
Figure 3.9: Sensitivity study for the varying greyness and constant perimeter Lagrange multiplier

Regarding the objective, it is shown that as the perimeter constraint becomes more and more dominating, such as in figure 3.9b, the performance of the design drops. If the perimeter constraint is not active such as figure 3.10d the performance is best, since this is an unconstrained design.

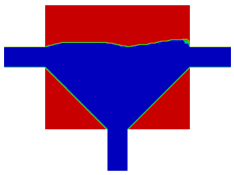
Secondly, the behaviour is shown as the perimeter weight factor λ_{perim} is increased. If the value of λ_{perim} is chosen too small as seen in figure 3.10e it can be seen that this constraint hardly has any effect, and the design is still curved. By increasing λ_{perim} the opposite behaviour from changing λ_{grey} is found. Therefore we can conclude that the ratio between the two values is important in finding the right design.



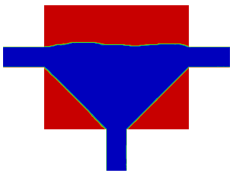
(a) Lagrange multipliers: $\lambda_{perim} = 5e-11$ and $\lambda_{grey} = 1e3$. Objective Function: $J = 0.0001222$



(b) Lagrange multipliers: $\lambda_{perim} = 4e-10$ and $\lambda_{grey} = 1e3$. Objective Function: $J = 0.0001504$



(c) Lagrange multipliers: $\lambda_{perim} = 6e-10$ and $\lambda_{grey} = 1e3$. Objective Function: $J = 0.00014$



(d) Lagrange multipliers: $\lambda_{perim} = 8e-10$ and $\lambda_{grey} = 1e3$. Objective Function: $J = 0.0001767$

(e) Lagrange multipliers: $\lambda_{perim} = 5e-9$ and $\lambda_{grey} = 1e3$. Objective Function: 0.004183

Figure 3.10: Sensitivity study for the constant greyness and varying perimeter Lagrange multiplier

3.3. TOTAL SENSITIVITY WITH VOLUME, PERIMETER AND GREYNESS CONSTRAINT

In the previous sections, we introduced three new constraints; the volume and perimeter constraint supplemented with the greyness constraint. In this section, we recap on the implemented constraints and combine all constraints into one sensitivity. The volume constrained is implemented according to the Augmented Lagrange Multiplier Method (ALM) as shown in [55]. For the perimeter constraint, we only use the about the derivative of the perimeter. Since the ALM also requires information about the perimeter itself it is more complex to implement. For this reason, the perimeter and the greyness constraint have both been implemented with a straight forward weight factor. However, ideally, the should be implemented with the ALM also.

After implementing the volume constraint the following Augmented Lagrange equation:

$$L_{aug} = J + \int_{\Omega} (\mathbf{u}, q) R d\Omega - \lambda_{vol} c_{vol} + w c_{vol}^2. \tag{3.12}$$

In this equation, J is the objective function, and c_{vol} the volume constraint and λ_{vol} and w are the Lagrange multiplier and weight factor, respectively. At this point, the sensitivity is extended by including the perimeter and greyness constraint and this results in the extended Augmented Lagrange equation, as shown in equation 3.13.

$$L = J - \lambda_{vol} c_{vol} + w c_{vol}^2 - \lambda_{perim} c_{perim} + \lambda_{grey} c_{grey} \tag{3.13}$$

Taking the total derivative of the augmented Lagrange equation with respect to α, we find that the objective does not depend on alpha and vanishes, leaving us with the final sensitivity equation:

$$\begin{aligned} \frac{dL_{aug}}{d\alpha} = \mathbf{u} \cdot \mathbf{v} V + \int_{\Omega} (-\lambda_{vol} + 2 w c_{vol}) \frac{\partial c_{vol}}{\partial \alpha} d\Omega \\ - w_{perim} \frac{\partial c_{perim}}{\partial \alpha} - w_{grey} \frac{\partial c_{grey}}{\partial \alpha}. \end{aligned} \tag{3.14}$$

If we substitute the derivatives of the constraints from the previous section in equation 3.14 we find the following equation which represents the new total sensitivity:

$$\begin{aligned} \frac{dL_{aug}}{d\alpha} = \mathbf{u} \cdot \mathbf{v} V + \int_{\Omega} (-\lambda_{vol} + 2 w c_{vol}) \frac{\partial c_{vol}}{\partial \alpha} d\Omega \\ - w_{perim} \nabla \cdot \left(\frac{\nabla \alpha}{|\nabla \alpha| + \epsilon} \right) \cdot |\nabla \alpha| - w_{grey} \frac{\left(1 - 2 \frac{\alpha}{\alpha_{max}}\right)}{\alpha_{max}}. \end{aligned} \tag{3.15}$$

This total sensitivity is implemented in the OpenFOAM solver and can be used separately or combined, if necessary. However, it was already shown that finding the correct parameters for the perimeter constraint is a challenge. Choosing parameters is also challenging because it is different for every design and different flow parameters. If all constraints would be activated, it is also necessary to find the right distribution between the three, complicating the process further. Therefore it is advised to use only one of the two constraints, or use the volume constraint to find an initial design and later activate the perimeter constraint to reduce the perimeter of the manifold. The latter method was applied for the three-dimensional optimisation, and the results are presented in chapter 4.

3.4. VERIFICATION OF THE OPENFOAM SOLVER

In this section, we compare the results of the new OpenFOAM solver w.r.t. previous published results by [10] and [55]. The former implemented Stokes flow and the latter included Navier-Stokes and turbulence in the adjoint equations. As discussed earlier the initial solver was not able to converge to practical enclosed manifolds for Stokes flow due to the subtle sensitivities. However, with the newly implemented volume constraint, we can now compare the results with Borrvall & Petersson. For turbulent flow, the solvers had stability issues finding enclosed solutions without the volume constraint, as the Reynolds number goes up this effect becomes stronger.

3.4.1. STOKES FLOW VERIFICATION

To verify the solver for Stokes flow, the design domain is given by Borrvall & Petersson as shown in 3.11 was recreated in OpenFOAM. The dimensions are assumed to be in centimetres, i.e., the inlet and outlet are $0.02m$. The inlet velocity is then given to be $0.01m/s$. For Stokes flow, the Reynolds number must be much smaller than one, and therefore we choose the kinematic viscosity to be $1m^2/s$. These two parameters result in a Reynolds number of 0.0002 which satisfies $Re << 1$ and is thus in the Stokes flow domain. Stokes flow is valid when the viscous forces are much higher than inertial forces, which means the Reynolds number is minuscule. Since the inertial forces are negligible, the convective term in the Navier-Stokes equations becomes zero, and thus we end up with the Stokes equations. Finally, the given volume fraction that is used for the design is set to 25% according to the paper by Borrvall & Petersson.

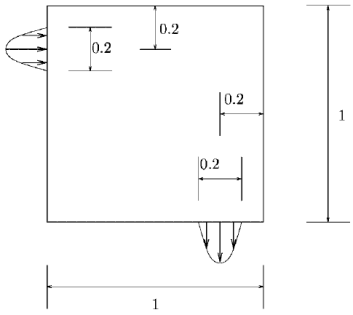


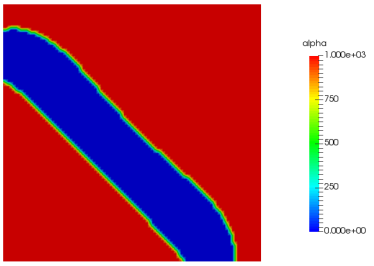
Figure 3.11: Design domain for elbow (Source: Borvall [10], 2003)

OPTIMAL DESIGN WITH OPENFOAM

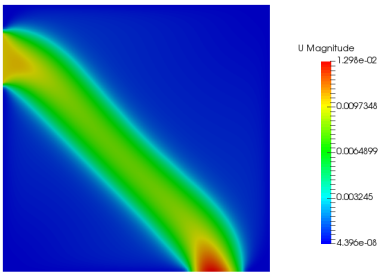
The optimal design found by Borrvall & Petersson is shown in figure 3.12a and in figure 3.12b the optimal geometry found by OpenFOAM is shown. In OpenFOAM, the red region means low porosity (no fluid flow) and the blue area high porosity (fluid flow). In figure 3.12c the velocity field is shown which illustrates the magnitude of the velocity in the new pipe. It can be concluded that this new design is very similar to the design found by Borvall and Peterson [10].

This solution was found in 711 iterations in only 74 seconds (Intel Core i5-4210 CPU @ 1.7-2.5 GHz) with a mesh of 10000 finite volume elements.

(a) Elbow: found by Borrvall & Petersson



(b) Elbow: optimal 'elbow' result for to Stokes flow

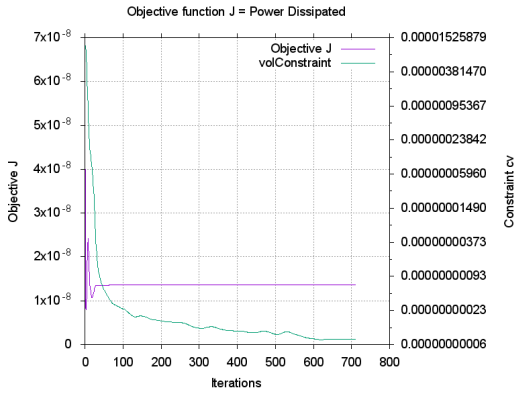


(c) Elbow: velocity magnitude

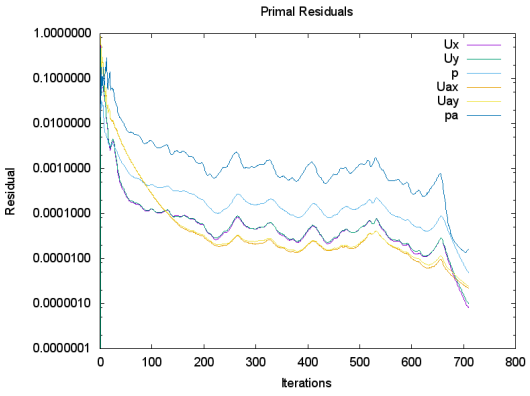
Figure 3.12: New design with the use of OpenFOAM

OBJECTIVE AND RESIDUALS

The objective used in [10] is minimum power dissipation. However, not all parameters are known. Therefore the absolute value may be different, and a comparison is not useful regarding the objective. In figure 3.13a we see the objective value and volume constraint value plotted for each iteration. As expected the constraint converges towards zero, and the objective is minimised which is essentially converged after 100 iterations. In figure 3.13b the residuals for the primal and adjoint velocity and pressure are shown, which are converged after 711 iterations according to a user defined criteria.



(a) Elbow: objective function and constraint value



(b) Elbow: residuals for the primal and adjoint velocity and pressure

Figure 3.13: New design with Stokes flow

3.4.2. NAVIER-STOKES FLOW VERIFICATION

In the previous section, the solver was verified for Stokes flow by recreating the same design that Borvall & Petersson found. The next step is to check the results for Navier-Stokes flow i.e. larger Reynolds numbers. In this case, the design domain is given in [55] and shown in figure 3.14 is used. This manifold distributes the flow from one inlet to two outlets. The Reynolds number of Re = 7000 is based on the inlet height and the viscosity. The algorithm is allowed to use 30% of the design domain. The mesh consists of 400x400 elements with a zero initialized Lagrange multiplier for the volume constraint.

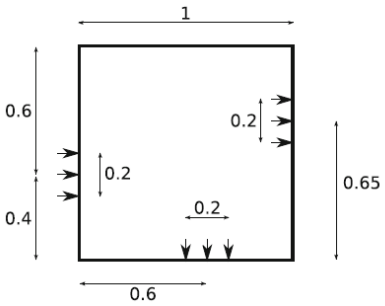
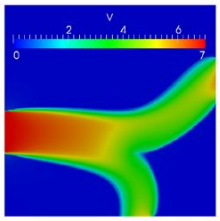


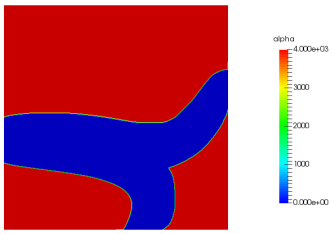
Figure 3.14: 2D manifold

OPTIMAL DESIGN WITH OPENFOAM

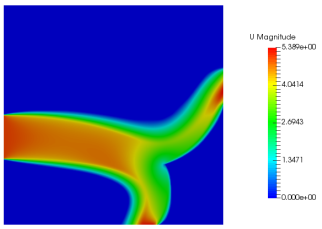
After 1500 iterations (57 minutes) the optimal shape shown in figure 3.15 and is rather similar to the result found in [55]. As the parameters of the problem at hand are not exactly known it is again impractical to compare the objective function graphs. Nonetheless, it can be seen in figure 3.16 that the objective and residuals converge.



(a) Manifold: optimal design found by Papoutsis



(b) Manifold: optimal design found in OpenFOAM

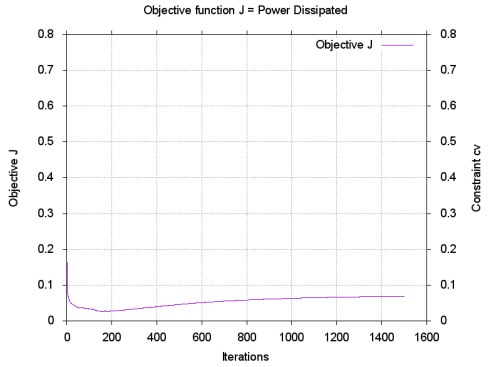


(c) Manifold: velocity magnitude

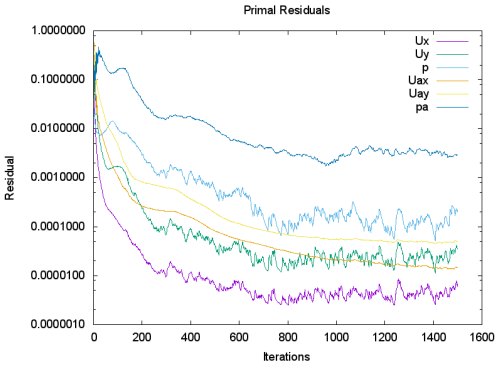
Figure 3.15: New design with the use of OpenFOAM

OBJECTIVE AND RESIDUALS

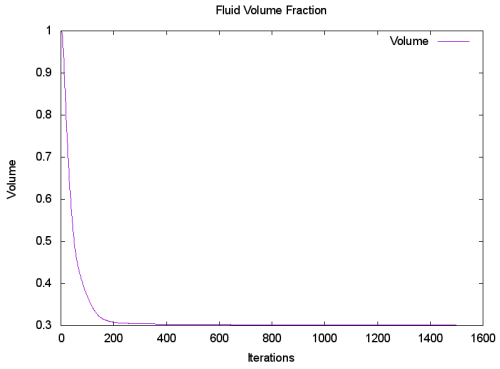
It is interesting to note that the final design as shown in figure 3.15 does not change after 700 iterations. However, the objective function is still converging to a given value. This is a result of the SIMPLE algorithm that uses non-converged solutions for each subsequent step. As a result, the actual flow solutions are still being solved until they reach a converged state, and thus the objective scales with these solutions. For example in figure 3.16d the turbulence residuals are still converging, and this will have an effect on the power dissipation.



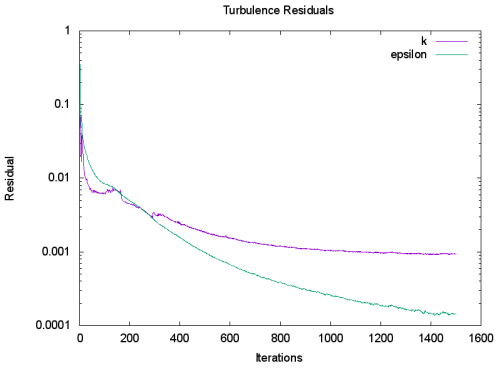
(a) Manifold: objective function and constraint value



(b) Manifold: residuals for the primal and adjoint velocity and pressure



(c) Manifold: volume fraction of the deisgn domain



(d) Manifold: residuals for the primal turbulence

Figure 3.16: Numerical figures showing convergence

For both Stokes and Navier-Stokes the solver appears to yield proper results as compared to current literature. Accompanied by the well-converged solutions of the solver it is safe to assume that the current implementation can be practically implemented as long as the user is aware of the limitations.

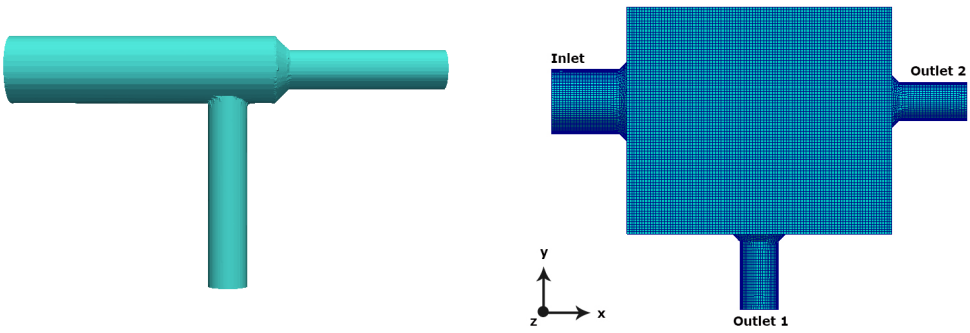
4

3D Topology Optimisation for Navier-Stokes Flow for Additive Manufacturing

Two dimensional topology optimisation is efficient for validating and testing the new constraints due to the low numerical cost. However, in practice, the results are not always usable because it is not possible to translate the 2D topology optimisation geometry to a 3D model for additive manufacturing. Therefore, it is necessary to find a way to generate stable 3D results in OpenFOAM. Subsequently, it is important to prepare these geometries for additive manufacturing regarding overhang. As discussed in the literature survey, internal cavities cannot be supported with sacrificial support material because it cannot be removed afterwards. In this chapter, the starting point for the 3D topology optimisation approach will be introduced, and the subsequent steps towards a printable design will be discussed.

4.1. Design Domain and Boundary Conditions

In the three-dimensional case, the conventional tee manifold as shown in figure 4.1a is used as the benchmark case. It is assumed that the conventional manifold that would be manufactured as an internal manifold i.e. holes drilled in a solid piece of material, similar to hydraulic manifolds. In this case, topology optimisation and additive manufacturing are expected to yield the best results with respect to the benchmark scenario. In figure 4.1b the full hex mesh of the design domain is shown, which is created in HyperMesh. For post processing, it is important to use a structured hex mesh, which can be utilised for overhang reduction purposes that will be discussed in chapter 4.4.



(a) Conventional internal tee manifold

(b) Design domain meshed with HyperMesh

Figure 4.1: Design domain for 3D manifold

The flow parameters are chosen such that the flow and pressure drop is within the specifications of the available flow and pressure sensors for testing. Furthermore, the geometric dimensions are selected such that the tee manifolds are not trivially manufactured with SLM i.e. large diameters. Aforementioned can be used to prove the post processing possibilities to produce a manifold that would usually collapse under its weight. In table 4.1 the parameters for the design domain are summarized.

Table 4.1: Parameters for the manifold for test set-up

Parameter	**Value**
Inlet diameter	0.017 m
Outlet diameter	0.01 m
Inlet velocity	1.2 m/s
Volume flow	16.34 l/min
Fluid	water
Dynamic Viscosity	1e-6 m^2/s
Density	1000 kg/m^3
Minimal Pressure drop	20 $mbar$

4.2. 3D Topology Optimisation Result

The design domain consists of 1.540.810 elements, and the simulation took 15 hours and 29 minutes for 1500 iterations. However, it can be seen in figure 4.3a that the solution was converged after 500 iterations. The same laptop CPU was used as for the 2D simulations. The resulting topology optimised geometry is shown in figure 4.2. The manifold is an unconventionally shaped tee, that is optimised with respect to minimum power dissipation. Moreover, the volume constraint was set to 25% of the design domain.

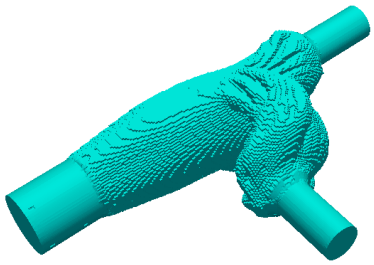
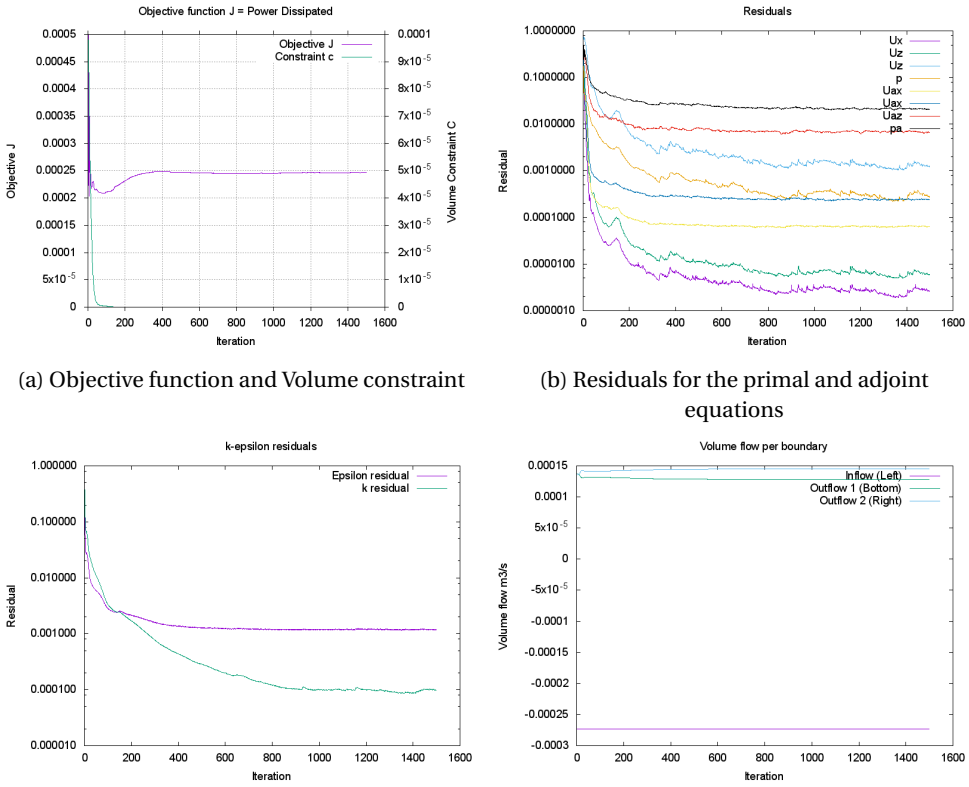


Figure 4.2: Topology optimised geometry found with OpenFOAM

To verify convergence, it is necessary to look at two different parameters. Firstly it is important to check if the primal, adjoint and turbulent equations are converging properly, by looking at the residuals. The residual values directly measure the error in the solution. It essentially measures the imbalance of a conserved variable in each element, meaning that a lower residual value represents a more precise solution. In [41] 1e-4 is considered loosely converged and 1e-6 is considered tightly converged. However, these values can differ per code or problem. Secondly, it is essential to see how the objective function and the constraint behave from the start. The objective is expected to converge to a certain optimal solution, and the volume constraint must converge to zero. The residuals, objective and constraint are shown in figure 4.3.

In figure 4.3a the objective function is shown to converge to a value of 0.00021 [J/s], and the volume constraint converges to zero, as expected. The residuals are shown in figure 4.3b and a distinct difference between the primal and adjoint values is observed as they are solved by different equations. The primal equations tend to converge to a lower residual and are thus more precise. The residual values of the pressure and z velocity are a few orders magnitude larger than the x and y velocity. Since pressure is solved differently, it is expected that the residuals differ from the velocity residuals. However, the z residual difference may be unexpected, but this same behaviour is seen if a different solver is used on the same design domain. Therefore, it may be inherent to OpenFOAM, and due to the fact, there is no inlet or outlet in the z direction.



(a) Objective function and Volume constraint

(b) Residuals for the primal and adjoint equations

(c) Residuals for the turbulent kinetic energy and dissipation

(d) Total (negative) volume flow at in and outlets

Figure 4.3: Topology optimisation results

4.2.1. NUMERICAL PARAMETERS

When switching from 2D to 3D simulations, multiple parameters are essential to the convergence of the problem. The most critical parameters are the relaxation factors. Each variable like pressure, velocity, and the turbulence can be relaxed with each successive iteration. Relaxation factors control the under-relaxation of the problem, which is a technique to improve the stability of the computation. It essentially limits the amount of change that is allowed for each variable per iteration. If a relaxation factor has the value of 1, it doesn't restrict the amount of change, and if the factor is 0, the variable cannot change at all. If the relaxation factor decreases, the under-relaxation will increase. The relaxation factors found to be stable (by trial and error) in this particular case are shown in table 4.2.

Table 4.2: Under-relaxation paramaters

Field	**Relaxation Factor**	**Value**
Design variable	α	0.3
Pressure & Adjoint Pressure	$p\&pa$	0.3
Velocity	**v**	0.95
Adjoint Velocity	**u**	0.6
Turbulence	$k\&epsilon$	0.6

It is also possible to choose the gradient and divergence schemes. Adjusting these schemes can yield different results. For the gradient scheme, the cell limited Gauss scheme is chosen, which can be given a value between 0 and 1. Similarly to the relaxation factor, this limiting factor clips each component of the gradient. Choosing the value one will increase the stability but reduce the accuracy. So it is a good practice to, reduce the limiting

factor as long as the simulation is stable. This method was chosen due to the relaxation freedom that was not possible for other methods. For every case and mesh size, it is necessary to tune these parameters to find a stable simulation.

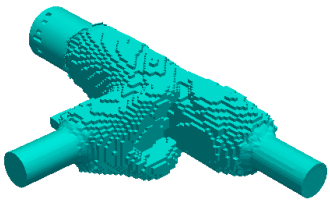
4.2.2. GRID CONVERGENCE

A common practice in CFD is to aim for a grid-converged solution. Grid convergence is a straight forward method that reduces the mesh size until a decent converged solution is found. As the numerical error with respect to the exact solution is dependent on the mesh size, and therefore the solution is expected to become more precise as the mesh is refined. For this purpose, the same domain is meshed in three different stages from coarse to fine.

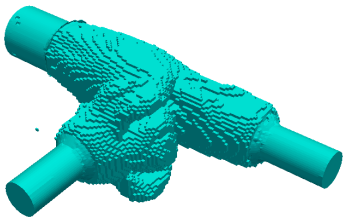
Table 4.3: Grid Convergence parameters

Grid	**Element size**	**# of elements**
Coarse	1e-3 m	209865
Medium	7.5e-4 m	476337
Fine	5e-4 m	1540810

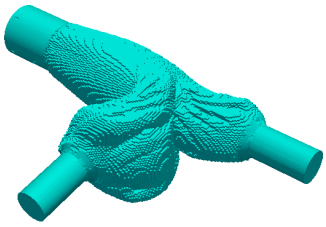
The mesh consists of 8 node hexahedral elements such that the results can be used for overhang removal as a post-process step. Tetrahedral elements would complicate the post-process step, but it should be possible to adjust the method for these type of elements. In figure 4.5 the geometry converges with increasing detail as the mesh is refined. Meshing elements finer than 5e-4 meter was not feasible with the computer power available for this research.



(a) Geometry coarse mesh



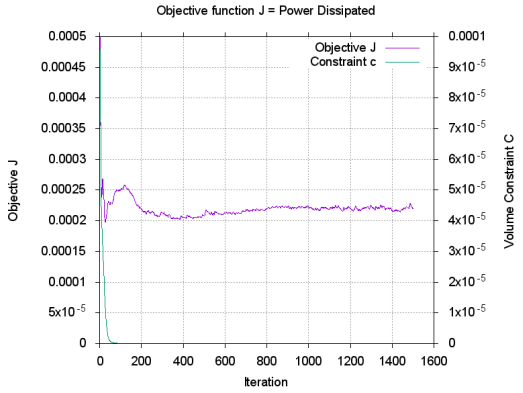
(b) Geometry medium mesh



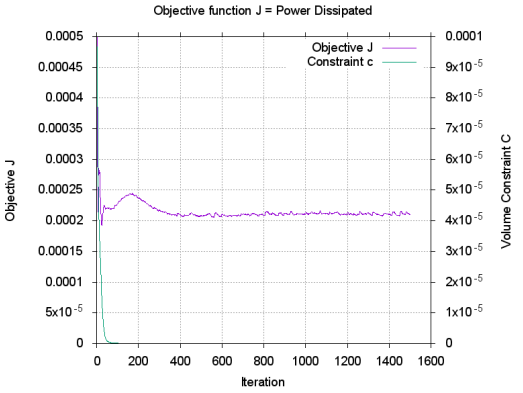
(c) Geometry fine mesh

Figure 4.4: Geometries for different mesh sizes

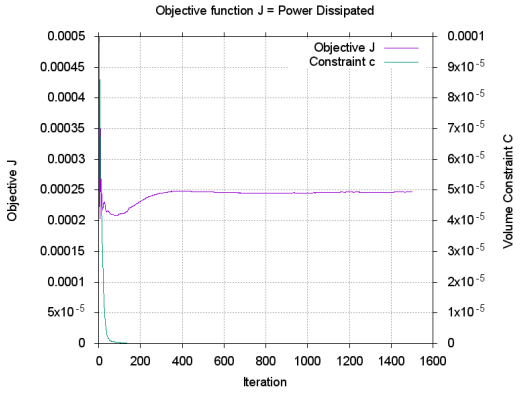
Regarding objective functions, we see that a coarse mesh shows a lower value for the objective function. This could be because smaller length scales are not taken into account for the coarser meshes. So the trend indicates that the objective value increases as the mesh is refined. However, the computational power prohibits us from further refining the mesh until we converge to a certain value. Therefore we cannot conclude what mesh size is necessary to converge as close as possible to the exact solution. But we can assume that coarse meshes underestimate the dissipated power and must be taken into account.



(a) Geometry coarse mesh: objective J = 0.000219



(b) Geometry medium mesh: objective J = 0.000210



(c) Geometry fine mesh: objective J = 0.000246

Figure 4.5: Geometries for different mesh sizes

Because the computational effort scales with n^3 i.e. the simulation takes eight times longer if the mesh size is reduced by a factor 2. The simulations took the following amount of time with the use of a standard laptop CPU, as shown in table 4.4. This simulation time represents 1500 iterations, but the objective was converged after about 600 iterations as shown in figure 4.5.

Table 4.4: Grid convergence simulation times and memory usage

Grid	**Simulation time**	**Mesh memory**	**Result memory (one time step)**
Coarse	24 *minutes*	40 *mb*	80 *mb*
Medium	241 *minutes*	94 *mb*	170 *mb*
Fine	915 *minutes*	300 *mb*	580 *mb*

4.3. 3D PERIMETER CONSTRAINT RESULTS

In the previous section, the 3D solutions were verified by looking at the residuals, objective and a grid convergence study. The same design domain and numerical settings are used as the foundation for the 3D results with the perimeter and greyness constraint. The working principle of the perimeter constraint combined with the greyness constraint is discussed in 2D in section 3.2. In this section, it is shown to be effectively and practically implementable for three-dimensional cases.

The first 500 iterations are included the volume constraint but without the perimeter constraint. Then the volume constraint is shut down, and the perimeter constraint is activated. Choosing these steps allows the perimeter constraint to start with a well-defined base design as previously shown in figure 4.2. It may not be necessary in every case, but it improved the stability of the perimeter constraint. The perimeter and greyness constraint are initialized with weight factors such that their sum is scaled to the initial sensitivity. After the initialization, the weight factors will gradually grow such that the effect of these weight factors can be shown.

The resulting objective function is shown in figure 4.6. It is shown that as the perimeter weight factor is increased, the objective function increases. Important to note is that the objective only increases slightly during the first 300 iterations. This means that the perimeter constraint yields the best result regarding performance loss. After 500 iterations the performance loss becomes so high that the weight gains are likely not justified.

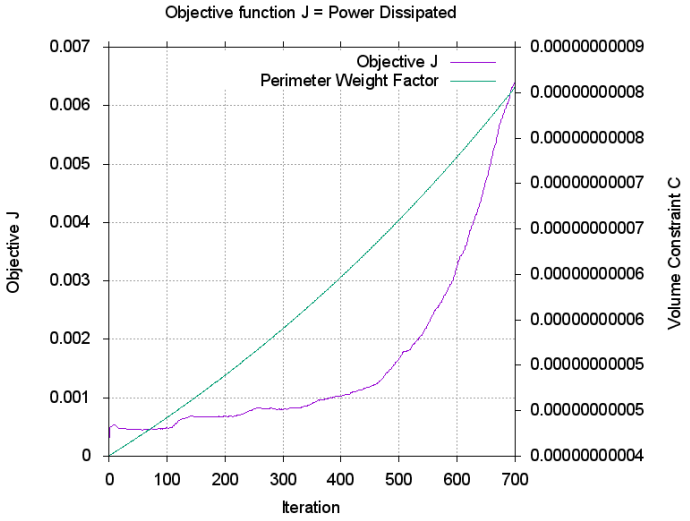
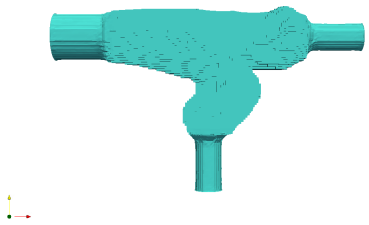


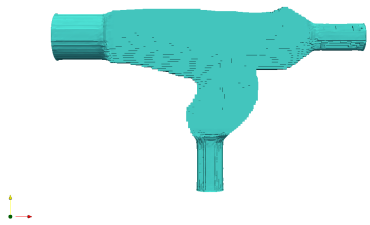
Figure 4.6: Objective function and the perimeter weight factor

In figure 4.7 the design of the manifold is shown for different time steps. The first figure shows the initial design that is found with the volume constraint. As the weight factor of the perimeter constraint is increased per iteration, it can be seen that it becomes more and more dominant. In the last figure, the design is almost exactly straight, which is expected as all the curvature is being penalised.

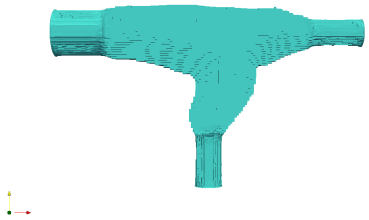
It can be concluded that the perimeter constraint (including the greyness constraint) perform very good for 3D problems. Moreover, it can be effectively used to reduce the weight of a manifold. It is suggested to use the perimeter constraint after an initial design is found with the volume constraint. Subsequently, the user should define how much the perimeter i.e. the weight of the manifold should be reduced, at the cost of performance.



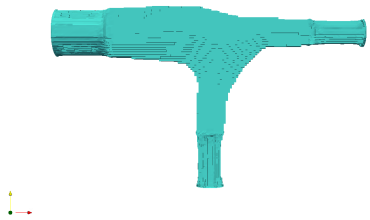
(a) Design after 500 iterations with the volume constraint



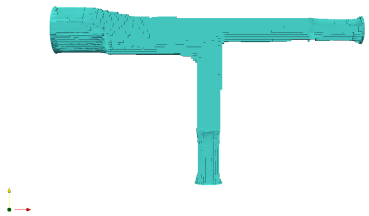
(b) Design after 700 iterations with perimeter constraint



(c) Design after 900 iterations with perimeter constraint



(d) Design after 1100 iterations with perimeter constraint



(e) Design after 1300 iterations with perimeter constraint

Figure 4.7: Design including perimeter constraint for different iterations and increasing perimeter weight factor

4.4. OVERHANG REMOVAL FOR ADDITIVE MANUFACTURING

After finding the optimal design with OpenFOAM, it is not necessarily manufacturable with SLM. This is due to the overhang that is present in manifolds, which cannot be solved with sacrificial support material, as it cannot be removed. In some cases, it is possible to put the entire manifold under 45 degrees such that there is no overhang. Unfortunately, this is not always the solution if the pipe or manifold is part of a larger structure. To solve the internal overhang problem, adjustments to the topology optimised geometry are necessary.

Current research is at the TU Delft [42] [69] [17] is looking to implement overhang constraints in the topology optimisation scheme, which would generate printable geometries and reduce support structures. As there is limited literature available on overhang constraints, and the implementation may require extensive C++ knowledge, this research will apply a post-processing step to remove overhanging regions. An algorithm is written for this purpose and is executed after the OpenFOAM topology optimisation is completed.

The main idea of this post-processing step is derived from [42]. The algorithm will look underneath each element to see if it is supported by material i.e. an element with $\alpha = 1$. The pseudo code is shown in algorithm 1. For the entire algorithm, we refer the reader to Appendix B.

Data: Create usable list of connected and below elements for each element
Data: Sort algorithm to sort List based on element coordinate (x,y,z direction)
for *All sorted elements* **do**
  if $\alpha \leq 1$ **then**
    Skip element
    else
      for *All elements below current element* **do**
        if *Element is supported* **then**
          Keep current α **else**
            $\alpha = 0$
          end
        end
      end
    end
  end
end

Algorithm 1: Pseudo code for post processing overhang algorithm

Due to the nature of the topology optimisation problem, the α field ranges from 0 to α_{max} such that the porosity is close to zero when $\alpha = \alpha_max$. Before the design field is used, it is transformed to field ranging from 0-1, which is done with the Logistic function

$$f(\alpha) = \frac{L}{1 + e^{-k(\alpha - \alpha_0)}}. \tag{4.1}$$

In the logistic function, L determines the maximum value of the function and k the steepness of the curve. This steepness means how 'black & white' the field becomes until it becomes a step function. Finally, it is necessary to determine the value of α_0 such that it is a reasonable value to decide if an element is solid or not. Since α ranges from 0 to 40000, in this case, it seems reasonable to take two-thirds of the maximum value $\alpha_0 = 26667$. Because a low α value still means that fluid can flow and it should be avoided that α_0 is too small.

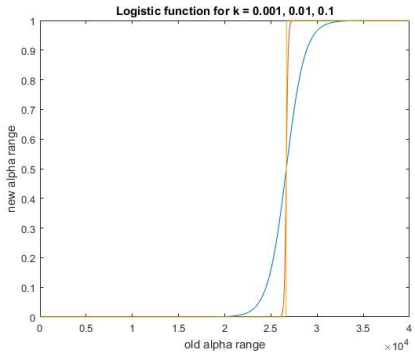
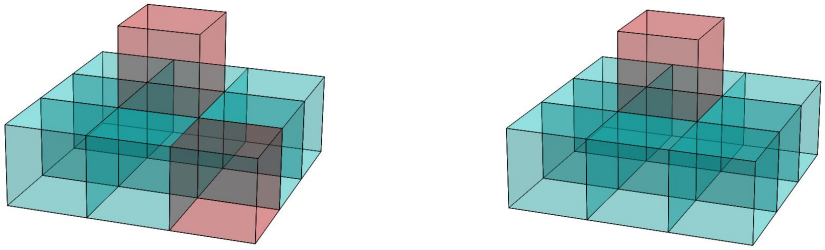


Figure 4.8: Logistic function converts the alpha field to a field ranging from 0-1

After looping over all elements sorted them and made a connectivity list it must be verified if an element is supported or not. In figure 4.9a an element is shown that is connected to nine elements below of which one element is filled with material ($\alpha = 1$ is shown in red). This means that regarding manufacturing overhang restrictions it is supported by the previous layer and is thus allowed. If the nine elements below are fluid (blue), it becomes apparent that there is no support for this element and it should thus be removed.

To define if an element should or should not be filled, it is possible to simply sum the values of α below the current element. However, that method could fill already fluid elements, which is not desired. Therefore, the following function is chosen to update the new α value for each element as shown in equation 4.2



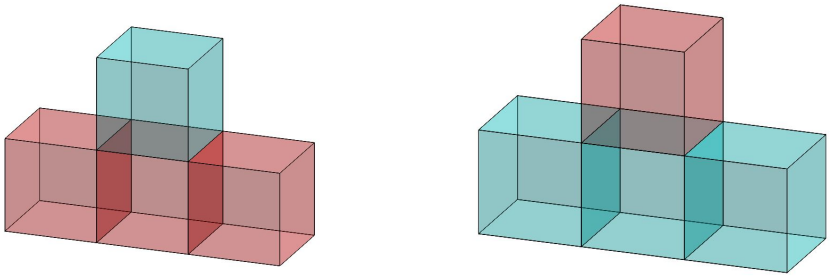
(a) Current element i is supported by a connected element

(b) Current element i is not supported by a connected element and should be removed

Figure 4.9: Difference between support and no support of a element

$$\alpha_i^p = min(\alpha_i^c, max(\alpha_j^p). \tag{4.2}$$

Where α_i^p is the updated printable field, α_i^c is the current field and α_j^p are the current printable alpha values of the elements below α_i^c. This function makes sure that in figure 5.5a the fluid element will not be filled with material. The maximum value on the right hand side of α_j^p is 1, the current α_i^c is 0, and thus the Min function will keep the value at 0. In the event that an element is not supported as shown in figure 5.5b the current α_i^c is 1 and the Max value of α_j^p is 0 and thus the Min function will set the printable α_i^p to 0.



(a) Current element i is supported

(b) Current element i is not supported

Figure 4.10: Difference between support and no support of a element

With this implementation, good results are found to modify topology optimised geometries. This makes the original design ready for additive manufacturing by making the manifold self-supporting in a user defined direction.

RESULTS OF THE OVERHANG REMOVAL

To show the results of this post-processing step, the manifold from figure 4.2 is again used as the initial design. The three different worst-case build orientations with maximum overhang are taken into account, and the algorithm for these orientations is applied. The worst case orientations in the x,y and z orientation are such that the overhang is perpendicular to the build plate i.e. 90-degree overhang. These different geometries can be investigated to show the performance of the overhang adjustments regarding dissipated power as compared to the topology optimised manifold.

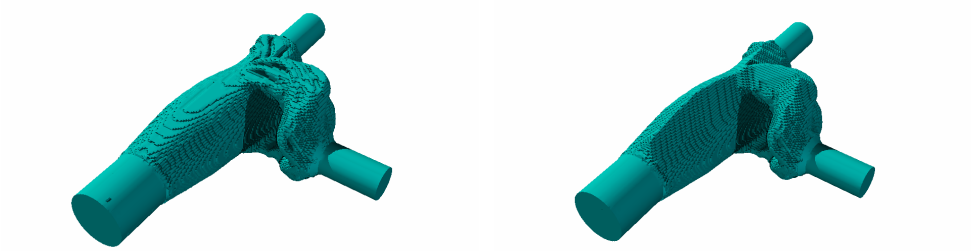
The resulting printable geometries can be seen in figures 4.11, 4.12, 4.13.

(a) Topology optimised design (b) Printable design in x direction

Figure 4.11: Printable manifold in the x direction

(a) Topology optimised design (b) Printable design in y direction

Figure 4.12: Printable manifold in the y direction



(a) Topology optimised design (b) Printable design in z direction

Figure 4.13: Printable manifold in the z direction

4.5. CFD SIMULATIONS OF SELF-SUPPORTING MANIFOLDS

The OpenFOAM topology optimised geometries are based on a porosity approach and work with a rather coarse mesh and wall. In the next step, the different designs are smoothed and re-meshed for further verification. In this section the three overhang adjusted designs are compared to the topology optimised tee as in figure 4.2 and conventional manifold as in figure 4.1a.

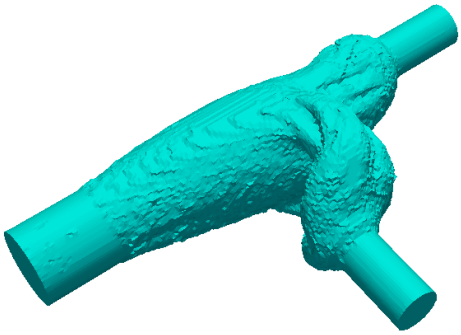


Figure 4.14: Smoothed geometry in OpenFOAM ready for verification

Since the geometry is smoothed and thus altered, different results can be expected. However, qualitatively we expect to see the similar distribution of pressure, velocity and turbulence fields.

4.5.1. SELF-SUPPORTING MANIFOLD PERFORMANCE RESULTS

The first observation that can be made is that the power dissipation is significantly lower than the topology optimisation simulation. This is expected to be due to the smoother geometry and the wall function. If the objective function values are compared, it can be seen that each adjustment performs differently from the optimised manifold.

In table 4.5 the difference regarding the objective value is expressed in percentages. It is clear that the Y adjustment performs worst and that the other two overhang adjustments perform significantly better. Nonetheless, each adjusted manifold is still superior with respect to the conventional manifold. Unfortunately, the topology optimised manifold can in some cases not be manufactured. As a result, the decreased performance is the price to be paid to manufacture it.

Table 4.5: Power dissipation difference with overhang adjustment

Geometry	Objective J	Difference
Conventional manifold	0.000219	
Original	0.000117	-46.5%
X overhang	0.000128	-41.5%
Y overhang	0.000158	-27.9%
Z overhang	0.000135	-38.4%

A second measure that could be of interest is how the flow distribution performs per outlet. To get insight into this matter, one can look at the mass flow at the inlet and outlets. These mass flow values are summarized in table 4.6.

Table 4.6: Mass flow per outlet

Geometry	**Mass flow outlet 1**	**Mass flow outlet 2**	**Ratio**
optimised	0.000132	0.000141	48.3%/51.7%
X overhang	0.000131	0.000142	47.9%/52.1%
Y overhang	0.000133	0.000140	48.7%/51.3%
Z overhang	0.000129	0.000144	47.2%/52.8%
Conventional manifold	0.000119	0.000152	43.9%/56.1%

It shows that the optimised and adjusted manifolds perform rather well in flow distribution as compared to the conventional manifold. In some practical cases, it could be important that the flow is well distributed, and then the optimised manifolds are a proper solution.

5

TOPOLOGY OPTIMISATION FOR MINIMUM WALL THICKNESS OF MANIFOLDS

So far the optimal fluid geometries were found in OpenFOAM and prepared for additive manufacturing. However, it is still necessary to find a way to optimise the local wall thickness around this fluid domain. To achieve this, a local wall thickness method is implemented by adding material where it is necessary and removing it where it is not. The internal pressure field is a good parameter to define where the walls should be thick and where they should be thin. In this chapter, it is shown how this pressure field can be used to find a locally optimised wall thickness.

5.1. STRUCTURAL TOPOLOGY OPTIMISATION WITH CFD PRESSURE FIELD

For more information about structural topology optimisation, the reader is referred to the literature survey for a brief introduction. The pressure field found by the CFD simulation can be used as input for a structural topology optimisation simulation. Utilising this method, the algorithm should add material where the pressure is high, and add less material where the pressures are low, resulting in an optimal, lightweight manifold. It should be noted that in the case of a simple tee manifold the operating pressure (2 MPa) will be far higher with respect to the local pressure variations. This means that the local deviating pressures are not having a big effect on the structural integrity of the manifold. However, there are certainly cases where local pressures can become large and then this method could yield good weight reduction results.

For this purpose, a python script was written to extract the pressure at each node on the boundary of the fluid domain, which can be found in Appendix D. The output of this script is a CSV (comma-separated values) file that can be implemented in Abaqus as a load condition. The internal, outward facing pressures need to be applied to the negative version of the fluid domain which is the structural design domain.

5.2. MINIMUM LOCAL WALL THICKNESS RESULTS

The design domain for the structural topology optimisation starts with a rectangular block of material. The mesh is build up from tetrahedron elements which are refined near the edge of fluid domain, and coarse near the outer walls of the design domain. The boundary conditions are defined as fully clamped on the sides where the in and outlets are present. This is done to simulate the connection to a different component.

The topology optimised geometry of the fluid domain is cut from this solid block such that the internal pressure can be applied on the same nodes. This pressure serves as a loading condition for the topology optimisation problem. The design domain, as created in Abaqus, is shown in figure 5.1.

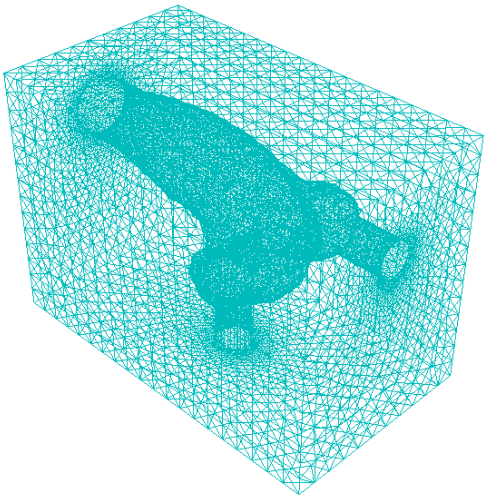
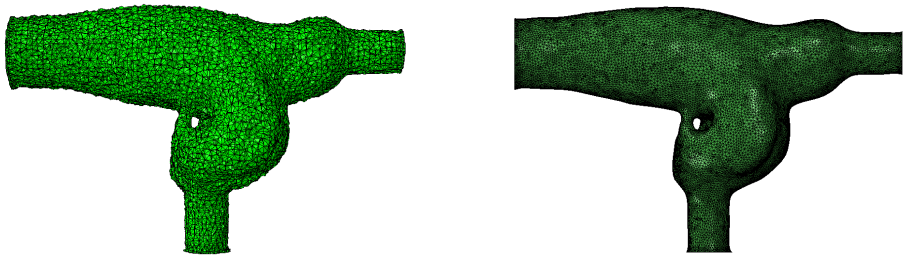


Figure 5.1: Design domain for structural topology optimisation

For this topology optimisation problem, the SIMP method is used which is implemented in Abaqus Tosca and is briefly discussed in the literature survey. For the structural topology optimisation problem, the strain energy is chosen as the objective function, which is a measure of the global displacement. A volume constraint (2.5% of the total domain) is also applied which can be arbitrarily chosen to fulfil a particular wall thickness, hence a safety factor.

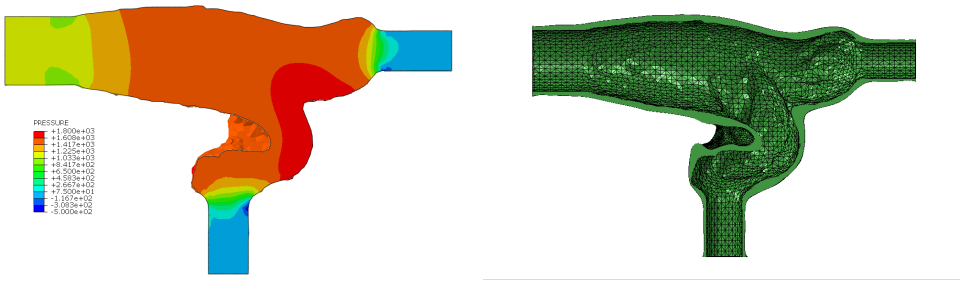
The resulting wall is very coarse because the solution consists of intermediate density values. Similar to the fluid topology optimisation this is called a grey solution i.e. not black % white. Hence, the result needs to be smoothed before the design can be used for manufacturing. It is evident that this step changes the geometry without looking at the performance, as a result, the performance may decrease as smoothing occurs. This smoothing step should be done with care and evaluated after the smoothing takes place. The difference before and after smoothing can be seen in figure 5.2.



(a) Non smoothed result (b) Smoothed result

Figure 5.2: Topology optimised walls in Abaqus

After smoothing, the final geometry is a manifold with local wall thicknesses, based on the internal pressure field. The wall thickness and the corresponding pressure can be seen in figure 5.3. It can be seen that at lower pressure regions such as the in- and outlets show a thinner wall. Important to note is that the smoothing step can change the wall thickness, as well as a different smoothing parameter, may increase or decrease the wall thickness over the entire domain.



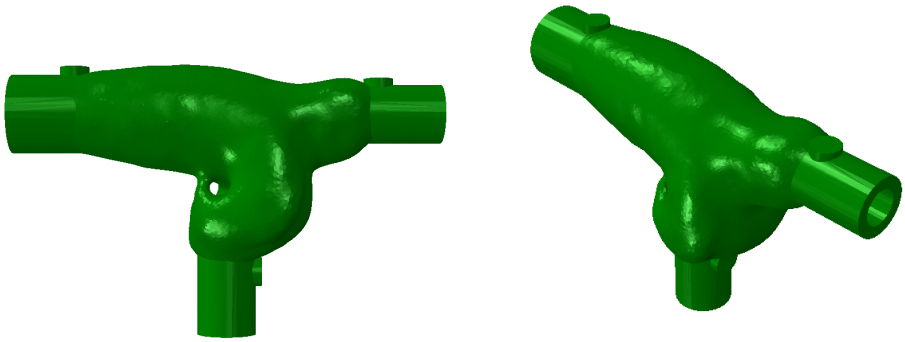
(a) Pressure field in Abaqus (b) Topology optimised cross section

Figure 5.3: Topology optimised cross section cut in the z plane

5.2.1. GEOMETRY PREPARATION FOR MANUFACTURING

The geometry found in the previous section not convenient to use for a test set-up. Hence, some post processing changes are necessary. First of all, the material is added at the in- and outlets which can subsequently be removed by conventional methods, such that the tolerances are fit for tube connections. The second adjustment is to add material at the locations where the pressure is measured. At these locations, a hole needs to be drilled such that small tubing can be connected to the pressure sensor. This is done to increase the structural integrity during the testing which is discussed in the next chapter.

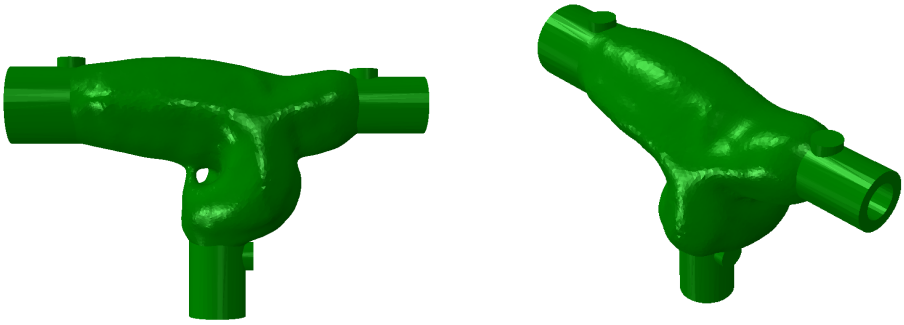
The final design of the topology optimised manifold is shown in figure 5.4. The overhang adjusted design in the z direction is shown in figure 5.5.



(a) Top view of the topology optimised geometry

(b) Iso view of the topology optimised geometry

Figure 5.4: No overhang: local wall thickness found with the structural topology optimisation



(a) Top view of the z overhang adjusted geometry

(b) Iso view of the z overhang adjusted geometry

Figure 5.5: Z overhang: local wall thickness found with the structural topology optimisation

6

Manufacturing and Testing of the 3D Topology Optimised Manifolds

The theoretical part of this thesis has so far been presented to the reader. The next step is to look at the practical side of manufacturing and post processing a printed part such that it is ready for use. Finally, two tests have been executed to look at the real performance of the manufactured manifolds. In this case, the optimised manifold, the x, and z build direction manifolds are taken into account. Due to limited space on the build plate, only three manifolds were printed, and the y-overhang manifold was omitted.

6.1. Additive Manufacturing of the Final Manifolds

The resulting geometries from the topology optimisation are manufactured of the AlSi10Mg alloy. This was the available material at the time of printing, and thus there is no specific reason for using it. However, it was advantageous for post processing the material, since Aluminium is rather soft.

The manifolds are printed on an aluminium build plate in the selective laser melting printer SLM280 from SLM Solutions. The three manifolds i.e. original, x and z overhang manifold are printed in a different orientation such that the overhang adjustments can be tested regarding flow performance and print quality. The final prints are printed with external support as this was not taken into account. The resulting build plate is shown in figure 6.1

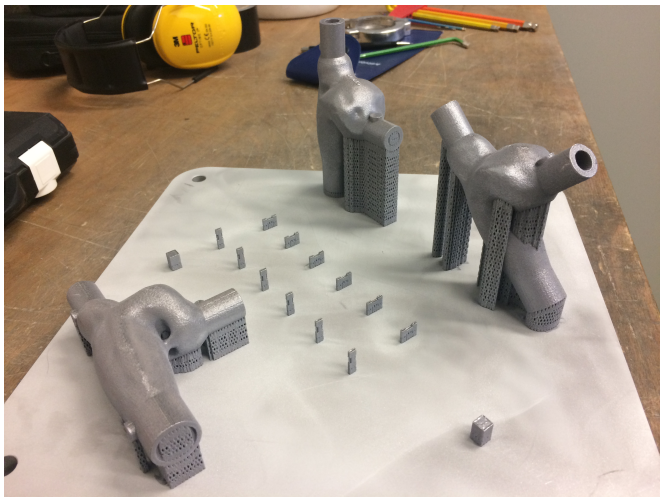


Figure 6.1: Manifolds on the build plate with external support material

After manufacturing the parts, several post-processing steps are necessary. The first step is to place the parts in the oven to reduce internal stress in the material by heating the parts up to a material-specific temperature. Subsequently, the support material is removed, which is still done manually or with a Dremel like tool.

For each manifold, the extra material must then be milled at the in- and outlets. Next, small holes were drilled for pressure the sensor. Due to the unconventional shape of the manifolds, post processing required more effort to support the printed part without damaging it. Therefore it is important that the post processing steps are taken into account beforehand. One could add some extra mounting points such that the part can be properly clamped and post processed. In figure 6.2 one manifold is clamped in the milling machine which was done at the Delft University of Technology.

Figure 6.2: Manifolds on the build plate with support material

A last mechanical post process step is to sand the surface of the manifold to get a homogeneous surface finish. This is done for both the visual aspect but also to remove semi-melted particles. Finally, each manifold can be seen in figure 6.3 and the manifolds in their printed orientation are shown in figure 6.4.

(a) Optimised manifold printed at an angle of 45°

(b) x overhang manifold, printed standing

(c) z overhang manifold, printed flat

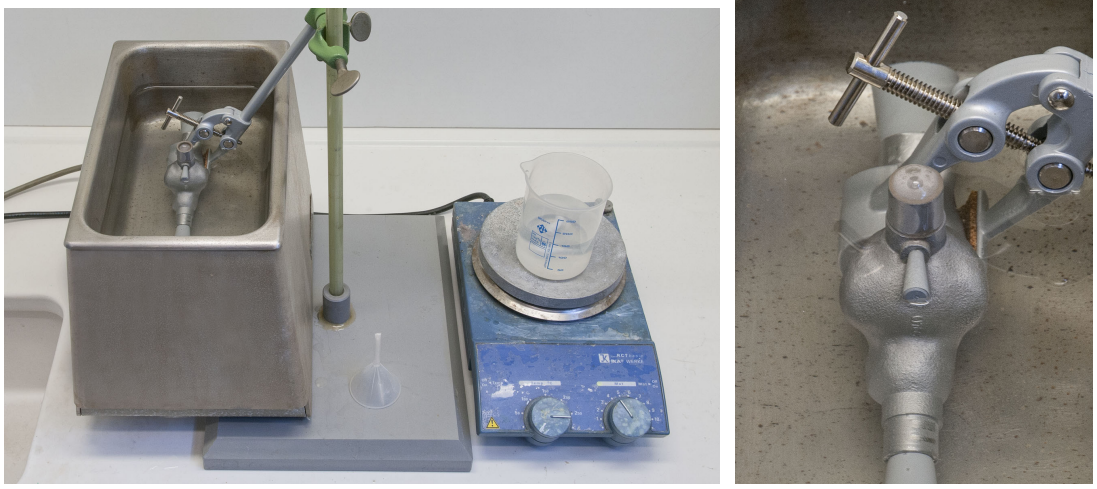
Figure 6.3: Final manifolds for the test set up

Figure 6.4: All three manifolds displayed in their printed orientation

6.1.1. CHEMICAL POLISHING OF THE INTERNAL SURFACE

After printing the manifolds, the exterior can be cleaned and sanded, improving the surface roughness. However, internally this is less trivial, and thus the surface roughness can be quite large. There are two reasons to polish the internal surface. First and foremost the performance of a manifold is better if the walls are smooth i.e. less friction. Secondly, any (semi) loose powder grains must be removed before using it in a flow system where it might damage other active components. As discussed in the literature review there are many methods to improve the surface of a printed part. Unfortunately, only a few methods are fit to be used in internal cavities. For this purpose, chemical etching is chosen to polish the internal surfaces of the manifold. Two other options were abrasive flow machining and electrolytic polishing. Unfortunately, no equipment was available for these larger sized manifolds.

Two methods are taken into account to apply the chemical etching process. The first option is to pump the etching reagent through the system, but a large pump and different tubing are necessary to achieve this. A second, more practical option is to close the manifold and fill it with the etching reagent. To improve the homogeneous effect of the chemical polishing process, the manifold is placed in an ultrasonic bath. This forces the etching reagent to keep moving internally.



(a) Test set-up with ultrasonic bath

(b) Manifold inside ultrasonic bath, filled with etching reagent

Figure 6.5: Etching (chemical polishing) set-up

Etching fluid from Keller's reagent was watered down to avoid excessive material removal. This is important because the local wall thickness is very thin at certain locations. The reagent consists of water, nitric acid, hydrochloric acid and hydrogen fluoride. The composition of the mixed reagent is shown in table 6.1.

Table 6.1: Reagent composition

Reagent	**Amount (ml)**	**Amount (%)**
H2O	190 ml	95%
HNO3	5 ml	2.5%
HCL	3 ml	1.5%
HF	2 ml	1%

For each manifold the following steps were carried out as follows:

- 5 minutes of ultrasonic etching
- 15 minutes ultrasonic cleaning with water
- Flushing with Ethyl Alcohol
- 15 minutes ultrasonic cleaning with Ethyl Alcohol

The composition of the reagent and the sequential steps are different for every case. Every material may need its type of reagent to achieve a certain amount of material removal and or surface roughness. A downside of chemical etching with respect to the other methods is that it tends to remove material everywhere, and thus not specifically reduce the surface roughness. As a result, the improved surface roughness is a side effect of the etching.

6.2. TESTING FOR PRESSURE DROP

After all the post processing steps the manifolds are ready for testing. The first test is a set-up to verify the pressure drop over each outlet. Due to limited equipment, the pressure drop per outlet is measured sequentially. It should be noted that the mass flow is fixed per outlet according to results from the CFD simulations. Otherwise, the flow distribution is mostly determined by the biggest pressure drop in the system, like the flow sensor in one of the branches. To avoid this effect, each branch should have the same components to cancel out differences in pressure drop, which was not feasible with the time and equipment at hand. For this test set up the following steps are carried out:

- Turn on pump and regulate mass inflow (three different volume flows)
- Switch on pressure measurement on outlet 1
- Measure the pressure at the outlet 1
- Turn off pump
- Switch on pressure measurement on outlet 2
- Turn on pump and regulate mass inflow (three different volume flows)
- Measure the pressure at the outlet 2
- Turn off pump

The test setup is shown in figure 6.6.

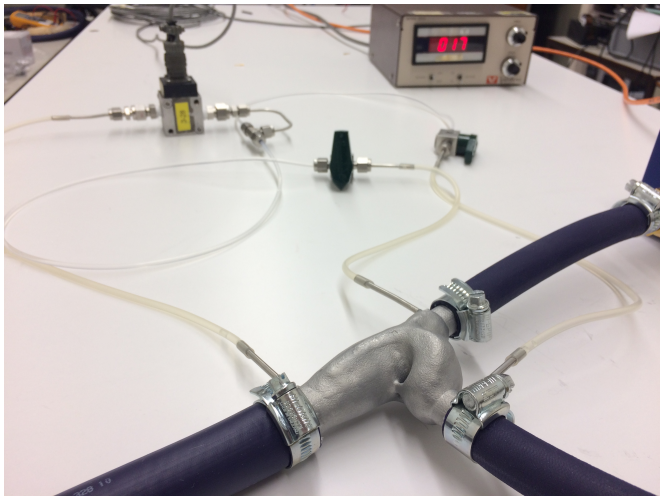
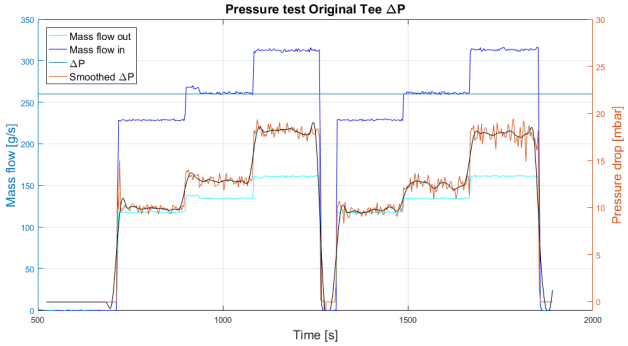


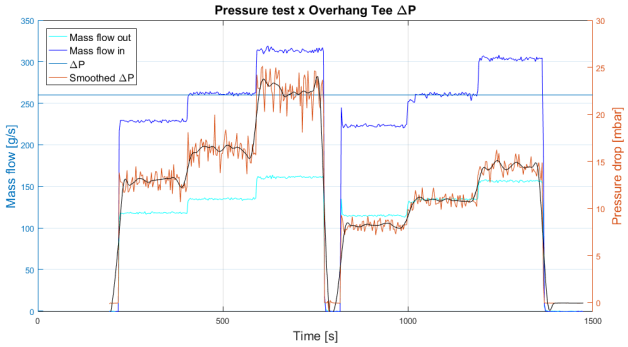
Figure 6.6: Pressure test set-up

6.2.1. PRESSURE DROP TEST RESULTS

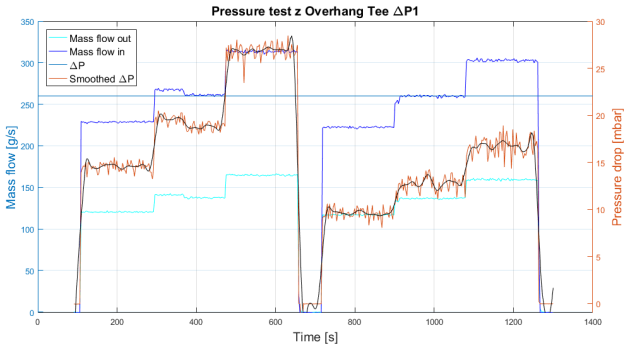
The results for each manifold are shown in the following graphs in figure 6.7. The left side of the graph measures the pressure at the first outlet, for three different velocities. On the right side of the graph, it shows the pressure at the second outlet. The different volume inflows were executed to see how the noise in the measurements evolved and to show the decrease in performance.



(a) Pressure test results for the optimised manifold



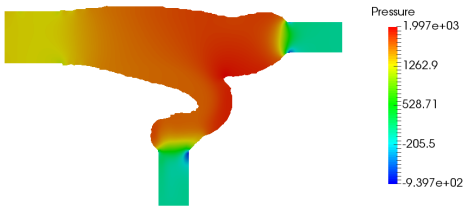
(b) Pressure test results for the x overhang manifold



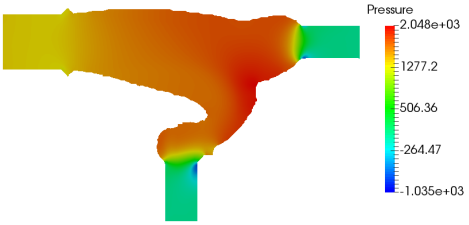
(c) Pressure test results for the z overhang manifold

Figure 6.7: xPressure test results for the z overhang manifold

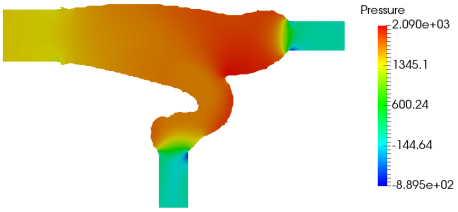
The first important observation is that the pressure drop at the first outlet (down) is experiencing a higher pressure drop for all three manifolds. Qualitatively this is as expected as compared to the simulations as shown in figure 6.8. The CFD simulations showed a decreased performance for the AM overhang adjusted manifolds. This effect is seen in both the pressure drop test and the pressure plots. The pressure drop is lowest in the optimised manifold and higher in both the adjusted manifolds. However, in absolute values, a difference is visible regarding pressure drop.



(a) Pressure plot from CFD simulation: optimised manifold



(b) Pressure plot from CFD simulation: x overhang manifold



(c) Pressure test results for the z overhang manifold

Figure 6.8: Pressure plot from CFD simulation: z overhang manifold

If the simulation results are compared with the test data, the following differences are found:

Table 6.2: Pressure drop at the outlets

manifold	ΔP_1 **simulation**	ΔP_1 **test**	**Difference**
Optimised	14.5 mbar	13 mbar	11.5%
X overhang	17 mbar	16 mbar	6.25%
Z overhang	18.2 mbar	17 mbar	7%

manifold	ΔP_2 **simulation**	ΔP_2 **test**	**Difference**
Optimised	12 mbar	12.5 mbar	4%
X overhang	12.3 mbar	11 mbar	11.8%
Z overhang	12.7 mbar	13 mbar	2.4%

Considering different assumptions along the way, a 10% difference is considered rather good. These differences are expected to be due to the following assumptions. First, it was shown that the solution is dependent on the mesh size and due to the lack of computational power, it was not possible to find grid convergence. Secondly, the turbulence model introduces assumptions in how the turbulence is modelled, choosing another method may yield different results. Lastly, in the CFD simulations the walls are assumed smooth, and thus the real manifolds are expected to perform differently due to the internal surface roughness.

6.3. TESTING FOR VOLUME FLOW DISTRIBUTION

The second test is executed similarly to the CFD simulations regarding boundary conditions. At the inlet, the volume flow is regulated and at the outlets, the water exits in 20L reservoirs. The latter simulates the zero pressure boundary condition at the outlet. This should give a good indication how well the flow is modelled in our simulation. However, a drawback of this test set up is that a significant transient start-up and the end effect is present. Unfortunately, the reservoirs were not large enough to execute the test for a long time to average out this effect.

6.3.1. VOLUME FLOW DISTRIBUTION TEST RESULTS

After running the test for 2 minutes, the weight of each reservoir can be measured. The ratio between the two masses gives us the volume flow distribution. Since this test is rather prone to transient start up and end effects the test is executed three times, and the results are averaged. The results are summarized in table 6.3 6.4 6.5.

Table 6.3: Topology optimised manifold flow distribution

Outlet	Test 1	Test 2	Test 3	Average
Outlet 1	15.16 kg	14.11 kg	14.70 kg	14.65 kg
Outlet 2	18.90 kg	19.70 kg	20.00 kg	19.53 kg
Ratio				42.8% / 57.2%

Table 6.4: X Overhang manifold flow distribution

Outlet	Test 1	Test 2	Test 3	Average
Outlet 1	15.51 kg	15.77 kg	15.12 kg	15.46 kg
Outlet 2	18.06 kg	17.53 kg	19.52 kg	18.37 kg
Ratio				45.6% / 54.4%

Table 6.5: Z Overhang manifold flow distribution

Outlet	Test 1	Test 2	Test 3	Average
Outlet 1	17.58 kg	14.77 kg	15.12 kg	15.82 kg
Outlet 2	18.74 kg	18.03 kg	17.82 kg	18.19 kg
Ratio				46.5% / 53.5%

The three manifolds perform qualitatively the same in the sense that more flow exits from the second outlet (straight). Moreover, the ratio between the in- and the outlet is different from the simulation i.e. performing worse as compared to the simulation. The first outlet (down) tends to have less volume flow with respect to the simulation data. If the volume flow ratios are compared and summarised it can be seen that the topology optimised manifold is performing significantly worse as compared to the other two manifolds as is shown in table 6.6.

Table 6.6: Difference between simulation and test data

Geometry	**Ratio Simulation**	**Ratio test**	**Difference**
Optimised	48.3%/51.7%	42.8% / 57.2%	24.8%
X overhang	47.9%/52.1%	45.6% / 54.4%	9.6%
Z overhang	47.2%/52.8%	46.5% / 53.5%	2.8%

A possible explanation for the difference in performance could be that internally; the walls have collapsed during the manufacturing. Because it is shown in figure 6.9 that the region in the red square is still significantly overhanging. Unfortunately, this was not visually reachable, and thus this hypothesis is not certain.

This overhanging region in the original manifold does show the importance of self-supporting manifolds as presented in this thesis. It becomes even more crucial if regions in a manifold are unreachable for visual inspection or small cameras like a borescope.

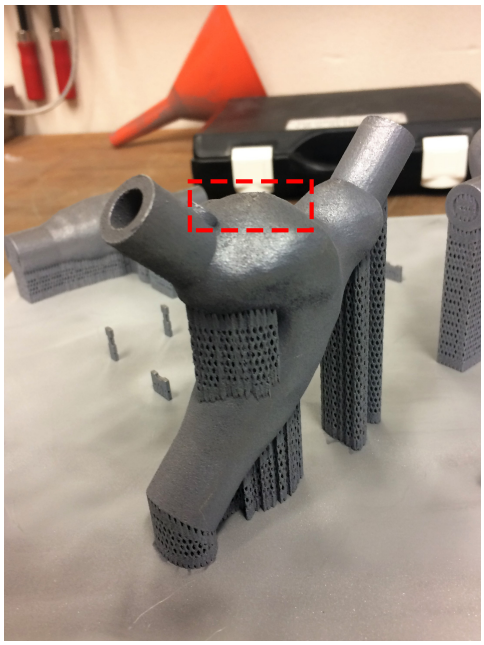


Figure 6.9: Topology optimised manifold in a 45 degree position

7

Conclusion and Recommendations

The entire topology optimisation design approach of manifolds, from unconstrained 2D topology optimisation to an optimised manufacturable 3D model, has been discussed in this thesis. The topology optimisation step was extended by incorporating weight minimising constraints. Subsequently, a post-processing step on the optimised model was carried out to adapt the geometry for SLM. Finally, a weight optimisation concerning local wall thickness was achieved with structural topology optimisation.

7.1. Conclusion

The goal of this thesis has been to investigate the topology optimisation approach for Naiver-Stokes flow manifolds that could be implemented practically and efficiently for additive manufacturing. In this concluding chapter, the research questions are answered as a result of this thesis research.

- *Can the performance (power dissipation) of manifolds be improved with continuous adjoint topology optimisation?*
  For the fluid flow topology optimisation method, it was shown that the continuous adjoint method is a very powerful method. For CFD calculations the number of elements (n) is enormous because a fine mesh is necessary to capture many length scales. The adjoint method reduces the computational effort from n+1 equations to just solving two equations; the primal and adjoint Navier-Stokes equations. Furthermore, it was shown that the 'one-shot' approach, which is inherent to the SIMPLE algorithm is very efficient for topology optimisation.

  Regarding the power dissipation objective, the topology optimised design yielded a reduction of 46.5% as compared to the conventional manifold, almost doubling the performance of the design. Furthermore, the mesh size is crucial for finding proper results of the CFD simulations. Hence a grid convergence study was executed to show that the geometry was converging to a particular shape and the objective function to a converged value. By refining the mesh multiple times, the objective value converged to a larger value than the initial coarse mesh. As a result, the mesh size should not be too large to avoid optimistic but wrong results. Moreover, reducing the mesh in half will increase the computational effort significantly and using a mesh that is unnecessarily fine should, therefore, also be avoided.

- *What methods can be used to minimise mass of manifolds manufactured with SLM?*
  The volume and perimeter constraint (including a greyness constraint) were implemented to control the amount of mass of the manifolds during the topology optimisation step. The volume constraint indirectly reduces the mass. Moreover, it is a very intuitive constraint to be practically used. The perimeter constraint, on the other hand, is directly proportional to the weight of the manifold but lacks the intuitive character of the volume constraint. The combination of these constraints gives the designer freedom in the topology optimisation design approach in the form of a trade-off between performance and mass reduction. Both these constraints ought to be weighted as compared to the performance of the manifold, choosing a more significant weight for the constraints will reduce the mass but also decrease the performance. The working principle of this perimeter approach is shown to be working

very well by finding straight perimeters, instead of the more organic shaped manifolds, and thus lighter manifolds. The perimeter constraint is complemented with a greyness constraint to avoid grey solutions, that is, intermediate porosity values. In a sensitivity study, these two constraints were shown to be working in a narrow spectrum. If the perimeter constraint was chosen too large with respect to the greyness constraint, the resulting design consisted of intermediate values. If the perimeter constraint is selected too small as compared to the greyness constraint the resulting geometry was not affected by the constraints at all. Consequently, the ratio between the two constraints should be well chosen with respect to each other, but their sum should also be in the order of magnitude of the original sensitivity. Important to note is that the parameters for this constraint can be significantly different for every topology optimisation problem and it can, therefore, be a time-consuming approach. The second method is a local wall thickness minimization based on the internal pressure field that results from the CFD simulation. The pressure field is subsequently used as a load condition for a structural topology optimisation step. This results in a local wall thickness for the manifold, such that it is thick where pressures are high, and thin where pressures are low. The resulting geometry is essentially ready for additive manufacturing. In this case, the benefits are problem-specific because it depends on the local pressure differences inside the manifold. For the simple tee manifold, the local pressure differences are very small as compared to the operating pressure.

- *How can the geometry of internal manifolds be adapted to meet SLM manufacturing requirements and how does this effect performance?*
  This research implemented an algorithm to remove the internal overhang after the topology optimisation step. The algorithm loops over all elements and sets the α value to zero if it is not supported, essentially creating a droplet-shaped cross section, that is, a 45-degree roof on top of the circular cross section. It is shown that this results in manufacturable geometries in different orientations without using sacrificial support material. The power dissipation of the x,y, and z build direction manifolds is reduced by 41.5%, 27.9%, and 38.4%, respectively. Thus even with the overhang adaptations, these designs still perform significantly better as compared to the conventional manifold. The manifolds were printed in three orientations i.e. the original design under a 45-degree angle, and the x and y build direction for the adapted manifolds. Visual inspections showed that the internal surfaces were without any sagging, which proves the feasibility of overhang removal approach.

- *Does the simulated performance of topology optimised manifolds approximate reality according to test set-ups?*
  Three manufactured manifolds have been tested to measure the internal pressures and the flow distribution. Both tests give information about the real performance of the manifolds but were not exact enough to use as a true validation of our simulation. For the pressure drop test, the mass flow was fixed at each outlet and measurements showed a difference in pressure drop of 3%-12% as compared to the simulations. The difference can have multiple causes such as in and outlet effects in the test set-up, inaccurate turbulence model or wall functions, the frozen turbulence assumption and the mesh size of our simulation. Secondly, the flow distribution was tested by connecting the outlets to two reservoirs and measuring the mass of each reservoir. The original optimised manifold showed a larger difference of 24% as compared with the simulation, which could be due to transient start up effects. However, the adjusted manifolds performed closer to the simulation data. A hypothesis for this difference is that the internal surface roughness of the original optimised manifold is rather rough due to overhang near the first outlet. However, no conclusive data proves these theories.

- *Is this topology optimisation approach for SLM practically applicable for the industry?*
  With the simulation and test data, it can be concluded that the topology design approach can be effectively implemented to find optimised and manufacturable designs for fluid flow manifolds. Besides the optimised performance, it is also possible to reduce the mass with the perimeter constraint and the local wall thickness optimisation. However, more detailed simulations and tests should be performed to validate the model for multiple geometries and Reynolds regimes. The practical side of this approach can still be optimised as many software packages are necessary from design domain to manufacturable design. During this project, HyperMesh, OpenFOAM, Abaqus, SolidWorks, Meshmixer and OpenFOAM were used to achieve the presented results. A common bottleneck is the STL file format that results from the topology optimisation step. As this file is build up as a surface consisting of many small triangles and not as geometry, it is a tedious process to adjust the topology optimisation results for manufacturing.

7.2. RECOMMENDATIONS

This research is only the beginning of a powerful design approach for optimising and manufacturing parts with AM. However, it can be improved on many fronts.

- First and foremost the adjoint method should be extended for full turbulence such that it can be used in all Reynolds domains. Work has been done on this front by [55] and can be used to implement in OpenFOAM.
- Secondly the mesh size is imperative in CFD to capture all length scales. There are methods available to refine a mesh while the simulation is running. The velocity or alpha field can be used to refine the mesh only at the boundary of the new geometry. With this method, it is also possible to implement wall functions during the topology optimisation. Some methods are available in OpenFOAM but need to be adapted for the topology optimisation solver.
- The current topology optimisation implementation does not take into account wall functions that are necessary for turbulent flow. This improvement can be combined with the active meshing near the boundaries because the wall function must be applied to the actual fluid to a solid boundary, which is unknown before the simulation starts.
- The post-processing step to remove overhang in the manifolds is effective but should be replaced with an active overhang constraint. This constraint should be implemented in the topology optimisation algorithm similarly to the volume and perimeter constraint. Since the post process step comes at a performance cost, an active overhang constraint might yield even better results. Past and current research at the TU Delft [42] [69] [17] already implemented basic constraints.
- The volume constraint is implemented with the Augmented Lagrange Method (ALM) which is a stable and accurate implementation. The perimeter and greyness constraint, on the other hand, are simply implemented with a weight factor. To improve these constraints, they could also be implemented with the ALM.
- Another important extension to this research is including the energy equation in the solver such that heat transfer can be taken into account for future topology optimisation. Many internal pipes and manifolds serve the purpose of cooling something. Combining Navier-Stokes flow with heat transfer may yield a very powerful design tool in 3D.
- The internal surface roughness is an important parameter concerning friction losses. Many surface treatments are available but hard to implement in internal cavities. A dedicated research could investigate different methods and how well they perform for different SLM alloys.
- For a true validation of the manufactured manifolds and their models, an exact test setup must be built to see the behaviour of the flow. These results can then be used to reiterate the turbulence model, wall functions and mesh size such that a better approximation can be achieved.

A

APPENDIX A: DERIVATION CONTINUOUS ADJOINT FOR DUCTED FLOWS

A.1. TOPOLOGICAL SENSITIVITY

This derivation follows the paper of C. Othmer [51] and may overlap with the literature survey, but more detail is given here. The continuous adjoint method is necessary to find the topological sensitivities of the optimisation problem. The Lagrange function which is presented in the literature survey is the start point for this derivation.

$$L = J + \int_{\Omega} (\mathbf{u}, q) R d\Omega \tag{A.1}$$

To find the sensitivities w.r.t. α, the design variable, the total variation of the Lagrangian function is taken, resulting in:

$$\delta L = \delta_{\alpha} L + \delta_{v} L + \delta_{p} L \tag{A.2}$$

As α is changed, there are also changes of the flow and pressure field. Hence, the total variation of L also includes contributions from the change in pressure and velocity. The direct method would use A.2 directly and solve once for each design variable, resulting in a computationally heavy method. However, the Lagrange multipliers can be chosen wisely such that the state variables vanish:

$$\delta_{v} L + \delta_{p} L = 0 \tag{A.3}$$

This allows us to simply calculate the sensitivities from the following equation:

$$\delta L = \delta_{\alpha} L = \delta_{\alpha} J + \int_{\Omega} (\mathbf{u}, q) . \delta_{\alpha} R d\Omega \tag{A.4}$$

This equation only requires to calculate cheap derivatives w.r.t. α and results in significant computational time reduction due to the uncoupled characteristic. This equation then yields the following sensitivity w.r.t. the porosity of cell i α_i:

$$\frac{\partial L}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \int_{\Omega} (\mathbf{u}, q) \frac{\partial R}{\partial \alpha_i} d\Omega \tag{A.5}$$

In this case the porosity is a variable that starts out as a full fluid domain, that is, $\alpha = 0$ and can be transitioned to a solid form i.e. $\alpha = \alpha_{max}$. Common objective functions have no dependency on α and therefore the first

term on the right hand side is zero. As α is only present in the primal equation it is only active in cell i and thus the Darcy term can be expressed as follows:

$$\frac{\partial R}{\partial \alpha_i} = \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} \chi_i \tag{A.6}$$

In this case χ_i is the characteristic function of cell i and it can be rewritten into equation A.5 as the final sensitivity function which is a dot product of the primal and adjoint velocities times the volume of cell i:

$$\frac{\partial L}{\partial \alpha_i} = \mathbf{u_i} \cdot \mathbf{v_i} V_i. \tag{A.7}$$

This shows that we only need the primal and adjoint velocities to compute the topological sensitivities for the optimisation problem. However, since we made the assumption in equation A.3 that the state variable contribution should vanish due to variations it is still necessary to derive the adjoint equations.

A.2. BOUNDARY CONDITIONS

In this section the boundary conditions are derived by deriving equation A.3, that is, $\delta_v L + \delta_p L = 0$. Which results in:

$$\delta_v J + \delta_p J + \int_\Omega (\mathbf{u}, q) \delta_v R d\Omega + \int_\Omega (\mathbf{u}, q) \delta_p R d\Omega \tag{A.8}$$

In the derivation of the incompressible Navier-Stokes equations with respect to velocity and pressure, the variation of the eddy viscosity is neglected. This is an important assumption that is commonly known as the Frozen Turbulence assumption. The resulting equation is shown in A.9.

$$\delta_v J + \delta_p J + \int_\Omega d\Omega \mathbf{u} \cdot ((\delta \mathbf{v} \cdot \nabla)\mathbf{v} + (\mathbf{v} \cdot \nabla)\delta \mathbf{v} - \nabla \cdot (2\nu D(\delta \mathbf{v})) + \alpha \delta \mathbf{v} - \int_\Omega d\Omega q \nabla \cdot \delta \mathbf{v} + \int_\Omega d\Omega \mathbf{u} \cdot \nabla \delta p = 0 \tag{A.9}$$

After decomposing the cost function on the boundary and interior domain, and integration by part, the final equation is formulated.

$$\begin{aligned} &\int_\Gamma d\Gamma \left(\mathbf{u} \cdot \mathbf{n} + \frac{\partial J_\Gamma}{\partial p} \right) \delta p + \int_\Omega d\Omega \left(-\nabla \cdot \mathbf{u} + \frac{\partial J_\Omega}{\partial p} \right) \delta p \\ &+ \int_\Gamma d\Gamma \left(\mathbf{n}(\mathbf{u} \cdot \mathbf{v} + \mathbf{u}(\mathbf{v} \cdot \mathbf{n}) + 2\nu \mathbf{n} \cdot D(\mathbf{u}) - q\mathbf{n} + \frac{\partial J_\Gamma}{\partial \mathbf{v}} \right) \cdot \delta v - \int_\Gamma d\Gamma 2\nu \mathbf{n} \cdot D(\delta \mathbf{v}) \cdot \mathbf{u} \\ &+ \int_\Omega d\Omega \left(-\nabla \mathbf{u} \cdot \mathbf{v} - (\mathbf{v} \cdot \nabla)\mathbf{u} - \nabla \cdot (2\nu D(\mathbf{u}) + \alpha \mathbf{u} + \nabla q + \frac{\partial J_\Omega}{\partial \mathbf{v}} \right) \cdot \delta \mathbf{v} = 0 \end{aligned} \tag{A.10}$$

This equation is satisfied for any $\delta \mathbf{v}$ and δp that satisfy the Navier-Stokes equations. The integrals over the domain describe the adjoint equations, that is, the adjoint Navier-Stokes equations as shown in equation A.11 and A.12

$$-2D(\mathbf{u}) = -\nabla q + \nabla \cdot (2\nu D(\mathbf{u})) - \alpha \mathbf{u} - \frac{\partial J_\Omega}{\partial \mathbf{u}} \tag{A.11}$$

$$\nabla \cdot \mathbf{u} = \frac{\partial J_\Omega}{\partial p} \tag{A.12}$$

Subsequently, the boundary conditions follow from the boundary integrals. Which result in the boundary conditions for the adjoint velocity and pressure as shown in equations A.13 and A.14, respectively.

$$\int_\Gamma d\Gamma \left(\mathbf{n}(\mathbf{u}\cdot\mathbf{v} + \mathbf{u}(\mathbf{v}\cdot\mathbf{n}) + 2\nu\mathbf{n}\cdot D(\mathbf{u}) - q\mathbf{n} + \frac{\partial J_\Gamma}{\partial \mathbf{v}} \right) \cdot \delta v - \int_\Gamma d\Gamma 2\nu\mathbf{n}\cdot D(\delta\mathbf{v})\cdot\mathbf{u} \tag{A.13}$$

$$\int_\Gamma d\Gamma \left(\mathbf{u}\cdot\mathbf{n} + \frac{\partial J_\Gamma}{\partial p} \right) \delta p = 0 \tag{A.14}$$

This derivation serves as the foundation for the continuous adjoint method that is used in this thesis. After this derivation, the equations must be specialised for ducted flow such that inlet, outlet and wall conditions are derived. For the boundary condition derivation, the objective function is important as this is the place were it explicitly appears. However, for the practical side of the derivation and objective implementation, we refer the reader to Othmer's work [51].

B

APPENDIX B: OVERHANG REMOVAL ALGORITHM FOR OPENFOAM

For the overhang removal as a post-processing step, we implemented the following C++ algorithm in OpenFOAM. In section 4.3 the working principle is discussed with the pseudo code. If the reader wants to understand or implement this code, we suggest to start with chapter 4.3 and then read the full code here. For this, to work, it is important that the mesh is built from hex elements only. If this method is to be used with different or mixed elements, this code must be adjusted.

Listing B.1: C++ script to create a manufacturable design

```
/*
Code written for overhang removal during this thesis project.
Author: Jeroen Verboom & Emiel van de Ven
Date: 1-5-2017
*/

#include "OFstream.H"
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// Initialize variables
//-------------------------------------------------
int maxElements = 9; // max number of elements that contain nodes + 1
int c_count = 0, lc_max = 0;
int pointIndex;
bool found;
double point, zPoint;
int elElMax = 10; // max elements under element + 1
int connectedEl;
double zOffset = 0.000325;
//-------------------------------------------------

// Initialize the point, face, cell and cellPoint lists
//-------------------------------------------------
// List of mesh points
const pointField& meshPoints = mesh.points(); // returns per point 3 coords: [nPoints x 3] list
// List of mesh faces
const faceList& meshFaces = mesh.faces(); // returns per face 4 nodes: [nFaces x 4] list
// List of mesh elements
const cellList& meshCells = mesh.cells(); // returns per cell 6 faces: [nCells x 6] list
// List of Points in Cells
const labelListList& cellPoints = mesh.cellPoints(); // returns per cell 8 points: [nCells x 8]
    list
//-------------------------------------------------

// Find the number of points and cells
```

```
//-------------------------------------------------
// number of points in the mesh
int nPoints = meshPoints.size();
// number of cells in the mesh
int nCells = meshCells.size();

Info << "Number of points: " << nPoints << endl;
Info << "Number of cells: " << nCells << endl;
//-------------------------------------------------

// Create empty lists
//-------------------------------------------------
// Create empty pointCell List
scalarList cellPointList(nPoints*maxElements,0);  // [nCells*9 x 1]
// Create empty z coordinate list
scalarList zCoordList(nCells,0); //[nCells x 1]
// Create empty connection list
scalarList cellConnectionList(nCells*elElMax,0);  // [nCells*10 x 1]
//-------------------------------------------------

//-------------------------------------------------
// Fill cellPointList [n1 el1 .. el8 n2 el1 ... el8 etc
// Fill zCoordinate list with center coordinates
//-------------------------------------------------

for (int i = 0; i<nCells; i++)
{
    //Loop over all points
   for (int j = 0; j<8; j++)  // Loops over the nodes of cell i: currCell is [1x8]
  {
      // current node from cell i
      point = cellPoints[i][j];

    // Storing nodes in cell point list
    //-------------------------------------------------
    pointIndex = cellPoints[i][j]*maxElements; // Gives the current point index of cell i

    cellPointList[pointIndex + cellPointList[pointIndex] + 1] = i;  // Storing cellnumber at
        pointindex in list

    // ??
    cellPointList[pointIndex] += 1;
    //-------------------------------------------------
  }
      // Extracting Z-coordinate of each element
    //-------------------------------------------------
    //Info << node << endl;
    zPoint = mesh.C()[i].component(2); // Height coordinate x = 0, y = 1 and z = 2
    //Info << zPoint << endl;
    zCoordList[i] += zPoint;
    //-------------------------------------------------
}

//-------------------------------------------------
// cellPointList and zCoordList are filled ready for next loop
//-------------------------------------------------


// Loop over all cells
for (int i = 0; i<nCells; i++)
{
  // Loop over all points
   for (int j = 0; j<8; j++)  // Loops over the nodes of cell i: currCell is [1x8]
  {

    // current node from cell i
      point = cellPoints[i][j];

    // loop over elements of node j of element i
    //-------------------------------------------------
    for (int k = 1; k<=cellPointList[point*maxElements]; k++)
```

```
108     {
109       connectedEl = cellPointList[point*maxElements + k];
110
111     // loop over elements of node point
112     //-------------------------------------------------
113      found = false;
114      for (int l = 1; l<=cellConnectionList[i*elElMax]; l++)
115                     {
116                         if (cellConnectionList[i*elElMax + l]==connectedEl)
117                         {
118                             found = true;
119                             break;
120                         }
121                     }
122         //-------------------------------------------------
123
124         // If found or element is current element, skip element
125             if (found == true || i == connectedEl)
126                 {
127                     continue;
128                 }
129         // Check if element is below current element, if so, add to list
130             if (zCoordList[connectedEl] < zCoordList[i]-zOffset)
131                 {
132                     cellConnectionList[i*elElMax + cellConnectionList[i*elElMax] + 1] =
                            connectedEl;
133                     cellConnectionList[i*elElMax] += 1;
134                 }
135
136     } // Closing Elements of Node Loop
137   } // Closing Node loop
138 } // Closing Cell Loop
139
140 //-------------------------------------------------
141 // Use cellConnectionList to loop over all elements and see if there is support
142 // 1. rewrite alpha field to 0-1 distribution (Logistics function)
143 // 2. Order zCoordinate list and create new indices vector
144 // 3. Loop over cellConnectList at for each sorted index
145 //  3.1 Loop over every connected element from 1-Connected Counter
146 //    3.1.1 Calculate new alpha based on min(alpha, max(alphaP))
147 //-------------------------------------------------
148
149 // variables for Logistic Function
150 int overK = 10;
151 int alpha0 = 2;
152
153 // create new alpha field > alphaLogistics (1-0 field)
154 forAll(mesh.C(), l)
155 {
156 alphaLogistics[l] = 1/(1+std::exp(overK*(alpha0-alpha[l])));
157 }
158 alphaLogistics.write();
159
160 // set alphaOverhang initially to the alphaLogistics field
161 alphaOverhang = alphaLogistics;
162
163 // Rewrite openfoam list to std::vector
164 std::vector<double> stdZcoordList (nCells, 0);
165 for (int sorti = 0; sorti<nCells; sorti++)
166 {
167 stdZcoordList[sorti] = zCoordList[sorti];
168 }
169
170 // Initialize empty zCoordIndex vector
171 std::vector<int> sortedZCoordIndex(stdZcoordList.size());
172 // Set first indice to zero
173 std::size_t n(0);
174 // Generate empty index vector from 0 to nCells
175 std::generate(std::begin(sortedZCoordIndex), std::end(sortedZCoordIndex), [&]{ return n++; });
176 // Sort the index vector based on the sorting of the z Coordinate list
177 std::sort(  std::begin(sortedZCoordIndex),
```

```
178             std::end(sortedZCoordIndex),
179             [&](int i1, int i2) { return stdZcoordList[i1] < stdZcoordList[i2]; } );
180
181 // print out sorted indecices based on z-height
182 //Info << "sortedZCoordIndex" << endl;
183 //Info << sortedZCoordIndex.size() << endl;
184 //for (auto v : sortedZCoordIndex)
185 //    std::cout << v << ' ';
186
187 // current alphaSupport
188 double alphaSupport;
189 // current max alpha support
190 double currMax;
191 // current sortedIndex from 0-nCells
192 int sortedIndex;
193 // current sortedConnectedIndex from 0-10*nCells
194 int sortedConnectedIndex;
195 // counter of how many cells bellow cell i
196 int connectCounter;
197 // lowest z coordinate in the model
198 double zMin = zCoordList[sortedZCoordIndex[0]];
199
200 // Loop over all cells
201 for (int sortj = 0; sortj<nCells; sortj++)
202 {
203   // nCells long ordered z from low to high gives index of first element
204   sortedIndex = sortedZCoordIndex[sortj];
205   // nCells*10 long   "     " gives index of Counter
206   sortedConnectedIndex = sortedIndex*elElMax;
207
208   // get the counter value of cell i, for loop purpose
209   connectCounter = cellConnectionList[sortedConnectedIndex];
210
211   // If cell has zero alpha i.e. fluid, we can skip this cell. if current elements has zero
        counter (no elements below) also skip.
212   if ((alphaLogistics[sortedIndex]) < 0.95 || (connectCounter == 0))
213   {
214   Info << "Skipping Cell"<< sortj << endl;
215   }
216   else
217   {
218     Info << "Changing Cell Value" << sortj << endl;
219     // Loop over all connected cells and below cell i, starting from counter + 1
220     for (int j = 1; j<=connectCounter; j++)
221     {
222         // get alpha value from the cell below cell i
223         alphaSupport = alphaLogistics[cellConnectionList[sortedConnectedIndex + j]];
224
225         // get max value from bottom points
226         if ( j == 1 )
227         {
228             currMax = alphaSupport;
229         }
230         else if ( currMax < alphaSupport )
231         {
232             currMax = alphaSupport;
233         }
234     }
235
236     // Set new alpha depending on min/max rule
237     alphaOverhang[sortedIndex] = std::min(alphaLogistics[sortedIndex],currMax);
238     alphaLogistics[sortedIndex] = std::min(alphaLogistics[sortedIndex],currMax);
239   }
240 }
241 alphaOverhang.write();
```

C

APPENDIX C: OPENFOAM C++ LIBRARY

In this appendix, we discuss OpenFOAM as an introduction for readers that want to use the topology optimisation code. First OpenFOAM is discussed in general and then show the practicalities of the adjoint solver.

C.1. OPENFOAM

OpenFOAM (OF) is the software of choice for this research because it is open source and it allows to change internal routines. Furthermore, it is the only available package with a continuous adjoint sensitivity solver, and therefore we will focus our attention on this program. OpenFOAM stands for Open (Source) Field Operation And Manipulation and is a highly symbolic C++ library for Computational Fluid Dynamics (CFD) problems. Since OF is open source and free to use under the GNU General Public Licence, it means that it can be used as a basis for in-house software. Furthermore, it comes with pre- and post-processing environments as is shown in figure C.1.

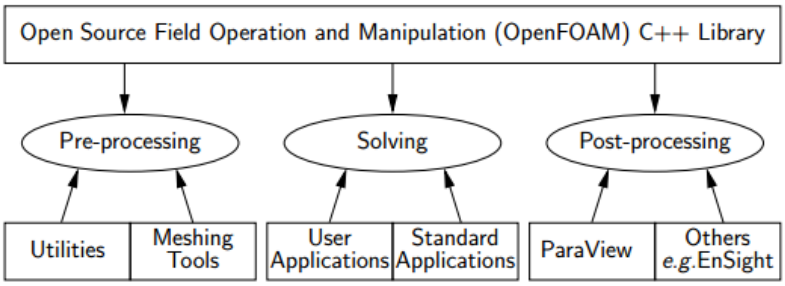


Figure C.1: OpenFOAM structure (OpenFOAM user guide)

C.1.1. OPENFOAM MESHING

For pre-processing, there are two native meshing tools namely blockMesh and snappyHexMesh which generate both hexahedron element meshes. However, since there is no GUI available blockMesh is rather tedious to use for complex meshes since every coordinate of every vertex needs to be written down by hand. SnappyHexMesh solves some of these problems as it allows the user to import a .stl file of the geometry and 'wrap' a mesh around it as shown in figure C.2.

Besides these native mesh methods, OpenFOAM can convert meshes from different programs such as Hypermesh, Fluent, CFX4 and Gambit. This gives more freedom to generate a complex mesh and use it to run a CFD simulation in OpenFOAM. For this research, we have used both blockMesh (2D) and Hypermesh (3D) to generate the design domains necessary for the topology optimisation.

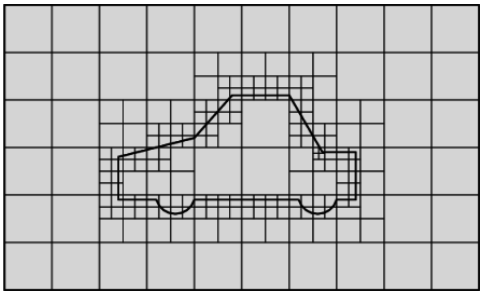


Figure C.2: snappyHexMesh principle

C.1.2. OPENFOAM SOLVING

Standard solvers are available in the OpenFOAM library which can be used and modified by users. This way multiple branches of solvers exist within OpenFOAM. For the purpose of adjoint topology optimisation, we use simpleFoam as the basis for the CFD solver. The Simple solver is used for steady and incompressible flow and uses the so-called Simple-algorithm. This solver is extended by Othmer [51] and is called the adjointShapeOptimizationFoam which will be used as a starting point for this project. There are no implemented constraints that can be used, and it is based on an old simpleFoam implementation and thus will be improved to fit our needs.

Moreover, many different solvers are available for incompressible transient problems, compressible problems, multiphase flow, combustion, heat transfer and many more. All in all, it is a very powerful open source package that can be used and extended to users requirements.

C.1.3. OPENFOAM CASE SET UP

A case (project) is set up as shown in figure C.3. It consists of text files and in some cases a mesh file from a different program like Hypermesh.

1. blockMeshDict
   Contains the geometry and mesh parameters.
2. controlDict
   Contains the input parameters such as time, step size and writing properties.
3. fvSchemes
   Contains the discretisation schemes for the finite volume method.
4. fvSolution
   Contains the linear equation solvers, tolerances and relaxation factors.
5. constant
   Contains the transport and turbulence properties if applicable.
6. polyMesh
   Generated files by blockMesh.
7. Time directories
   Files that contain the variable data for each time step.

It is important to note that if we talk about time steps for steady state solvers such as simpleFoam we talk about iterations since there is no time dependency in a steady state solver.

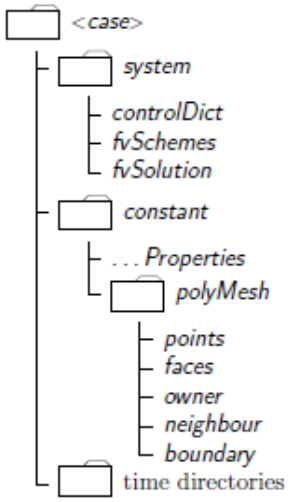


Figure C.3: OpenFOAM structure (OpenFOAM case setup)

C.1.4. OPENFOAM IMPLEMENTING CODE

Due to the symbolic nature it is relatively easily to adjust for different problems including topology optimisation. For example the following partial differential equation, the steady-state incompressible Navier-Stokes equation would be implemented as follows:

$$\nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p \tag{C.1}$$

Listing C.1: OpenFOAM implementation of a PDE

```
solve
(
fvm::div(phi, U)
- fvm::laplacian(mu, U)
==
- fvc::grad(p)
);
```

This is made possible by the custom name space Foam:: that is developed with OpenFOAM, so a lot of standard classes and utilities are available in this namespace. However, it is also possible to use the standard std:: namespace that comes with C++. The latter can be useful because there is more troubleshooting help online for the standard C++ than there is for OpenFOAM specific problems.

C.1.5. POST-PROCESSING: PARAFOAM

ParaFoam is a command that launches ParaView which serves as a GUI for data visualisation. When a case is solved, the output is opened in ParaView. There is a wide variety of possibilities within ParaView like vector plots, streamline plots, section cuts and much more. This post processor comes native with OpenFOAM and has everything we need for this thesis.

There are two other options to post-process data which can be done via de Command Line Interface (CLI) with standard utilities, and new utilities can be implemented by the user. The second option is EnSight which is a third party post processor for the simulation data.

C.1.6. OPENFOAM PROS AND CONS

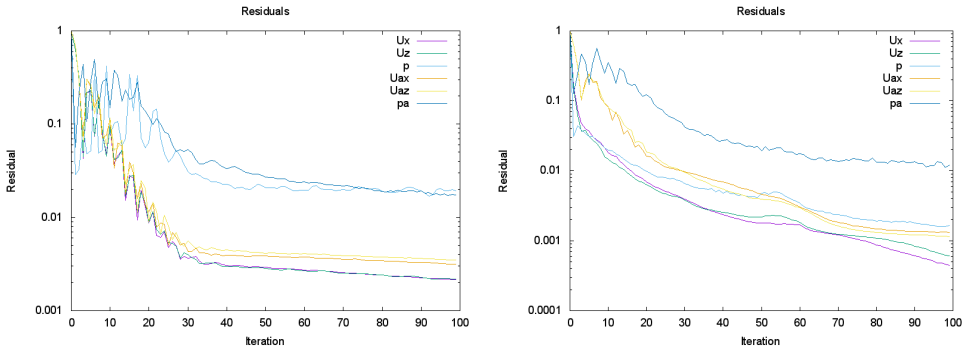
It can be concluded that OpenFOAM is a very powerful tool that is free to use by anyone and a lot of information is shared through the large community. The whole program is based on C++ which makes it a very fast and efficient code for CFD simulations.

The downside of OpenFOAM is that it has a rather steep learning curve regarding setting up and getting to know the underlying code. It is recommended that a new user has some C++ background before diving into the code. The native mesh utilities are not very user-friendly, and therefore it is also recommended to use a third party meshing software to avoid tedious meshing.

C.2. OPENFOAM CONTINUOUS ADJOINT SOLVER

The current solver implemented in OpenFOAM is called `adjointShapeOptimizationFoam` and is based on the incompressible Navier-Stokes solver `simpleFoam`. The solver essentially makes three calls. First, the primary solver is called to calculate the (primal) velocity and pressure, v and p, respectively. Secondly, it calls the adjoint solver to solve the adjoint equations for u and q, adjoint velocity and adjoint pressure, respectively. In the third call, it combines the primal and adjoint velocity into the topology sensitivities which it uses for the steepest descent algorithm such that the porosity α is updated.

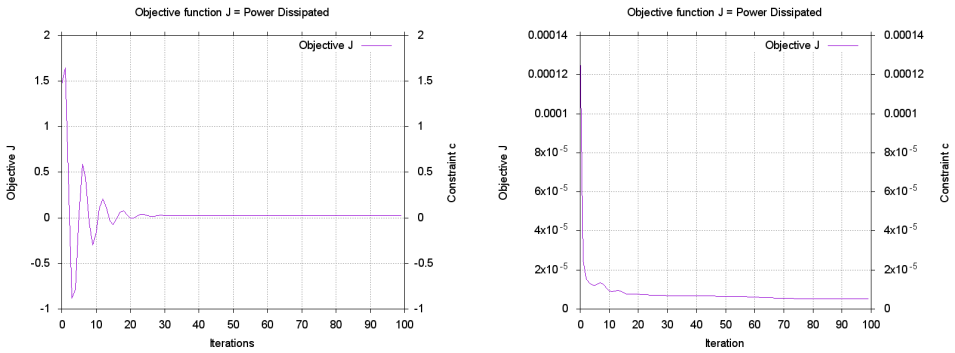
However early testing showed that the solver was very unstable at the beginning of a simulation resulting in diverging residuals and negative objective (power dissipation) values which can be seen in figure C.4 and C.5. These residuals and objective plots come from a simple elbow simulation with one input and output. It is obvious that negative power dissipated or power created is physically impossible. The new implementation results in smooth residuals and thus smooth pressure and velocity fields. From this point on we will only discuss the new implementation and will call it `adjustedAdjointSolver`



(a) Current adjointShapeOptimizationFoam implementation

(b) New implementation (expanded on simpleFoam)

Figure C.4: Residuals



(a) Current adjointShapeOptimizationFoam objective results

(b) New implementation (expanded on simpleFoam) objective results

Figure C.5: Objective function values

C.2.1. FILE STRUCTURE: ADJUSTEDADJOINTSOLVER

Like any other solver in OpenFOAM the main file is a C source file:
`adjustedAdjointSolver.C` which uses different header (.H) files with the #include command. The file structure necessary for the new solver is shown in figure C.6

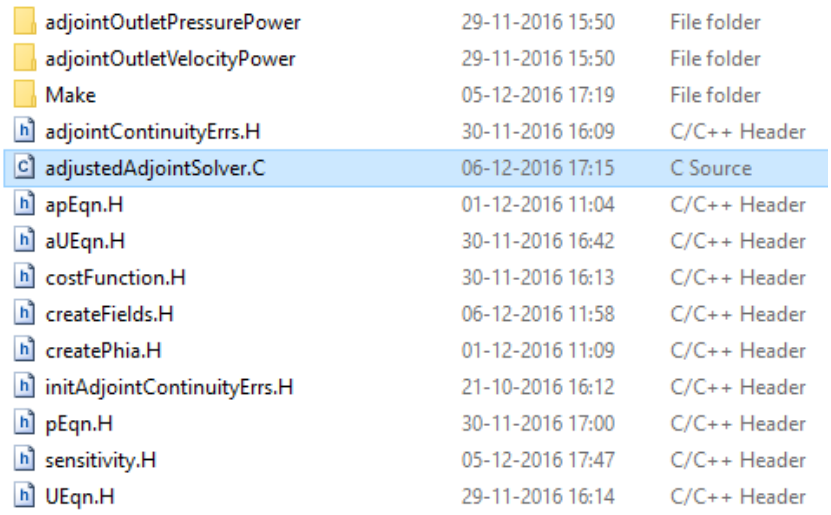


Figure C.6: File structure for adjustAdjointSolver

In the Make directory, there is an 'options' and 'files' document which are specific for each solver. The options document links to the header files necessary for the solver. In the files document, one needs to change the name of the executable to the name of your solver. The other files are briefly summarised in the following section.

C.2.2. HEADER FILES: ADJUSTEDADJOINTSOLVER

The adjoint directories are both the boundary conditions arising from the derivation of the continuous adjoint method. These boundary conditions can be called upon in any case and applied to the adjoint pressure and velocity boundary conditions. The other .H files are included in the main solver like a library.

I `createFields.H`
Initializes the necessary variables in terms of fields and scalars.

II `pEqn.H` and `UEqn.H`
Calculates the primal pressure and velocity according to the SIMPLE method.

III `apEqn.H` and `aUEqn.H`
Calculates the adjoint pressure and velocity according to the SIMPLE method.

IV `initAdjointContinuityErrs.H`, `adjointContinuityErrs.H` and `CreatePhia.H`
Are adjoint versions of the primal files.

V `costFunction.H`
Calculates the cost function value for plotting purposes.

VI `sensitivity.H`
Calculates the new sensitivity including volume, perimeter and greyness constraint

C.2.3. MAIN LOOP: ADJUSTEDADJOINTSOLVER.C

The main loop can be seen in listing C.2. The momentum and pressure equations are being solved using the SIMPLE algorithm (Semi-Implicit Method for Pressure-Linked Equations). When a steady-state problem is

being solved iteratively, it is not necessary to fully converge the linear pressure-velocity coupling, since the changes are no longer small between consecutive steps.

Listing C.2: Main SIMPLE loop of the solver

```
while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;


        zeroCells(alpha, inletCells);

        // --- Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.H"
            #include "pEqn.H"
        }

        // Cost Function
        #include "costFunction.H"

        // --- Adjoint Pressure-velocity SIMPLE corrector
        {
            #include "aUEqn.H"
            #include "apEqn.H"
        }

        // Correcting terms in SIMPLE loop
        laminarTransport.correct();
        turbulence->correct();

        // Calculate Sensitivity
        #include "sensitivity.H"

        laminarTransport.lookup("lambda") >> lambda;

        // Steepest Descent
        alpha +=
              mesh.fieldRelaxationFactor("alpha")
             *(min(max((alpha - lambda*newSens), zeroAlpha), alphaMax) -
                 alpha);

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }
```

D

Appendix D: Pressure Extraction Script for Minimum Local Wall Thickness

In this appendix, the Python script is presented that can be used to extract the pressure at each node. It uses the data from a CFD simulation as input for a topology optimisation step. The method and results are discussed in this thesis, but the script is distributed here for completeness.

Listing D.1: Python script to extract internal pressure from CFD simulation

```
#
    ____________________________________________________________________

## PACKAGES
#
    ____________________________________________________________________


import os
from abaqus import *
from abaqusConstants import *
from caeModules import *
from driverUtils import executeOnCaeStartup
from odbAccess import *
import csv
import numpy as np

#
    ____________________________________________________________________

## INPUTS
#
    ____________________________________________________________________


working_path     = 'C:\Temp\Tee_3D_Re10000_Smooth'
odb_name         = 'Tee_3D_Re10000_Smooth'
step1            = 'cfd'

odb_file     = openOdb(working_path + os.sep + odb_name + '.odb',readOnly=
    True)
odb_frame    = odb_file.steps[step1].frames
n_frames     = len(odb_frame)


# find node set
node_set     = odb_file.rootAssembly.nodeSets['NodeSet-1']
```

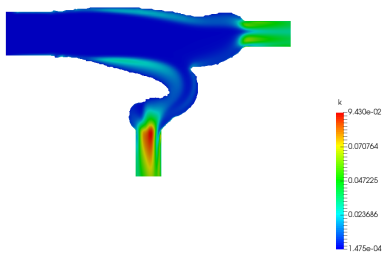
```
29
30 #
       ____________________________________________________________________________
31 ## EXTRACT DATA
32 #
       ____________________________________________________________________________
33 forLen = len(node_set.nodes[0])
34 coordTotalArray = [[0,0,0,0]]
35
36 for i in range(0,forLen):
37     temp = odb_frame[n_frames-1].fieldOutputs['PRESSURE'].getSubset(region=
           node_set).values[i].data
38     coords = node_set.nodes[0][i].coordinates
39     coordTotalArray = np.concatenate((coordTotalArray, np.array([[coords[0],
           coords[1], coords[2], temp]])), axis=0)
40
41 #
       ____________________________________________________________________________
42 ## SAVE DATA
43 #
       ____________________________________________________________________________
44
45 np.savetxt("PressureCoordinate.csv", coordTotalArray, delimiter=",")
46 \caption{Python script to extract pressure field from CFD simulation}
```

E

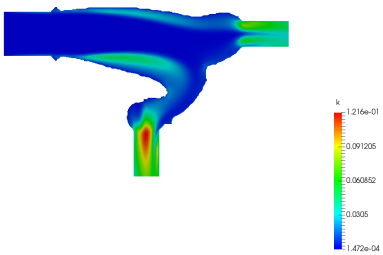
Appendix E: CFD Simulation Results of Self-Supporting Manifolds

In this appendix, the differences, due to overhang removal, are discussed. The differences are visualised with turbulent viscosity, velocity and pressure plots.

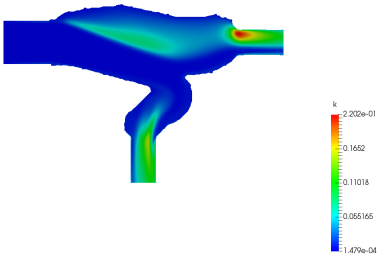
In figure E.1 the turbulent kinetic energy is plotted for each manifold. For laminar flow, the molecular viscosity is the driving diffusion term. But when the flow becomes turbulent the turbulent viscosity is added to model turbulence. Hence it is a virtual term. The turbulent viscosity is proportional to the kinetic energy squared. The different designs show significant different turbulent kinetic energy, which may be an important factor in the different performance values.



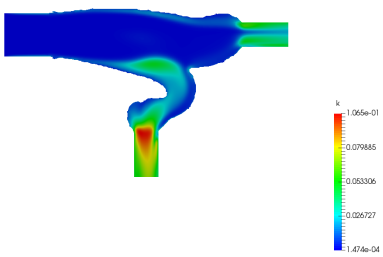
(a) Original manifold



(b) X overhang adapted manifold



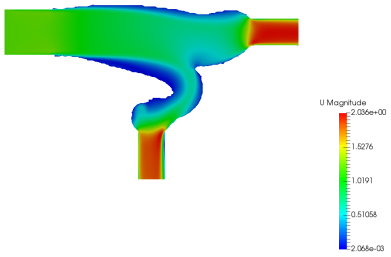
(c) Y overhang adapted manifold



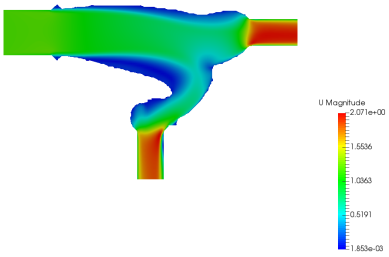
(d) Z overhang adapted manifold

Figure E.1: Turbulent kinetic energy differences due to overhang adjustments

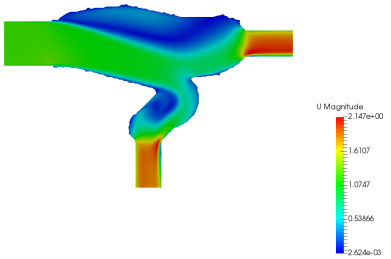
In figure E.2 the velocity plots show different flow patterns, but the magnitude is rather similar. These different flow patterns are likely the result of different flow distributions found during the testing phase.



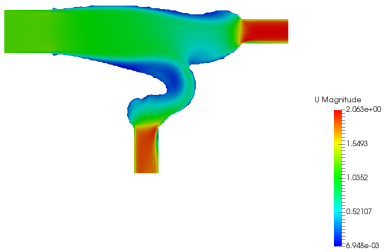
(a) Original manifold



(b) X overhang adapted manifold



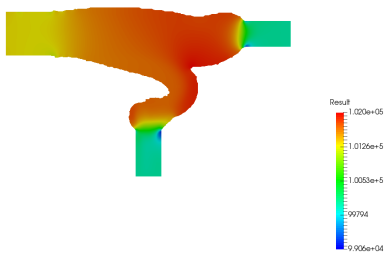
(c) Y overhang adapted manifold



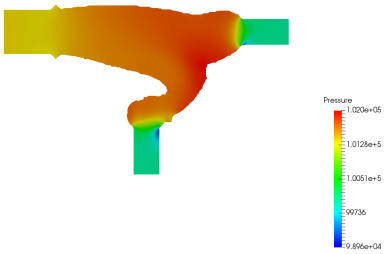
(d) Z overhang adapted manifold

Figure E.2: Velocity differences due to overhang adjustments

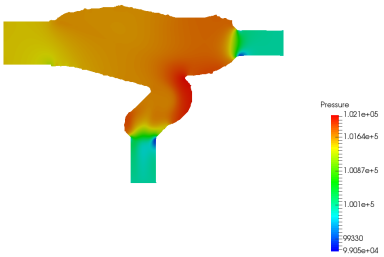
The pressure plots also show similar behaviour that the pressure is different in certain regions. This does show the principle of the different wall thicknesses based on the pressure field, which is different for each design.



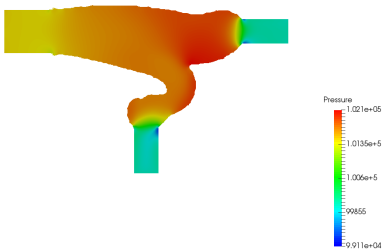
(a) Original manifold



(b) X overhang adapted manifold



(c) Y overhang adapted manifold



(d) Z overhang adapted manifold

Figure E.3: Pressure differences due to overhang adjustments

F

APPENDIX F: EXTENDED CFD LITERATURE SURVEY

F.1. EDDY VISCOSITY MODEL

In laminar flow, only the molecular viscosity is present, which is a physical property. The eddy viscosity model is based on the virtual turbulent viscosity, which is used to capture the diffusion of turbulent flow. The turbulent viscosity is modelled with two new partial differential equations for the turbulent kinetic energy and the turbulent dissipation rate as shown in equation F.1 and F.2, respectively.

$$\frac{\delta}{\delta t}(\rho k) + \frac{\delta}{\delta x_i}(\rho k u_i) = \frac{\delta}{\delta x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\delta k}{\delta x_j}\right] + P_k + P_b - \rho_\epsilon - Y_M + S_k \tag{F.1}$$

$$\frac{\delta}{\delta t}(\rho \epsilon) + \frac{\delta}{\delta x_i}(\rho \epsilon u_i) = \frac{\delta}{\delta x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon}\right)\frac{\delta \epsilon}{\delta x_j}\right] + C_{1\epsilon}\frac{\epsilon}{k}(P_k + C_{3\epsilon}P_b) - C_{2\epsilon}\rho\frac{\epsilon^2}{k} + S_\epsilon \tag{F.2}$$

These two equations are substituted in the Navier-Stokes equations through the turbulent viscosity term:

$$\nu_t = C_\nu \frac{k^2}{\epsilon}. \tag{F.3}$$

BIBLIOGRAPHY

[1] *Scripting User's Manual*. Abaqus.

[2] Allaire, Grégoire, Jouve, François, Toader, and Anca-Maria. Structural optimization by the level-set method. *Free Boundary Problems*, Jan. 2004. doi: 10.1007/978-3-0348-7893-7_1. URL http://dx.doi.org/10.1007/978-3-0348-7893-7_1.

[3] G. Allaire. A review of adjoint methods for sensitivity analysis, uncertainty quanti cation and optimization in numerical codes. *Ingenieurs de l'Automobile*, 2015.

[4] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 2004.

[5] G. Allaire, A. Karrman, and G. Michailidis. Scilab code manual, 2012.

[6] C. Aumund-Kopp and A. Riou. *Introduction to additive manufacturing technology*. European Powder Metallurgy Association (EPMA), 2015.

[7] N. Bakhtiary. A new approach for sizing, shape and topology optimization. *SAE International Congress and Exposition*, 1996.

[8] N. T. Basse. Turbulence intensity and the friction factor for smooth and rough-wall pipe flow. *X*, 2017.

[9] M. P. Bondsoe and O. Sigmund. *Topology Optimization: Theory, Methods and Applications*. Springer, 2003.

[10] T. Borrvall and J. Petersson. Topology optimization of fluids in stokes flow. *International Journal for Numerical Methods in Fluids*, 2003.

[11] D. Brackett, I. Ashcroft, and R. Hague. Topology optimization for additive manufacturing. *Solid Freeform Fabrication Symposium*, 2011.

[12] S. Bremen, W. Meiners, and A. Diatlov. Selective laser melting a manufacturing technology for the future? *www.laser-journal.de*, 2012.

[13] S. L. Campanelli, N. Contuzzi, A. Angelastro, and A. D. Ludovico. Capabilities and performances of the selective laser melting process. *New Trends in Technologies: Devices, Computer, Communication and Industrial Systems*, 2010.

[14] V. J. Challis. A discrete level-set topology optimization code written in matlab. *Structural Multidis*, 2010.

[15] *Simulation-Based Fluid Optimization To Accelerate Development Of Innovative Products*. Dassault Systems.

[16] J. D. Deaton and R. V. Grandhi. A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Struct Multidisc Optim*, 2014.

[17] A. M. Driessen. Overhang constraint in topology optimisation for additive manufacturing: a density gradient based approach. Master's thesis, TU Delft, 2016.

[18] X.-B. Duan, Y.-C. Ma, and R. Zhang. Shape-topology optimization for navier–stokes problem using variational level set method. *Journal of Computational and Applied Mathematics*, 2008.

[19] X.-B. Duana, Y.-C. Maa, and R. Zhangb. Shape-topology optimization for navier–stokes problem using variational level set method. *Journal of Computational and Applied Mathematics*, 2008.

[20] T. Furumotoa, T. Uedaa, T. Aminob, and A. Hosokawaa. A study of internal face finishing of the cooling channel in injection mold with free abrasive grains. *Journal of Materials Processing Technology*, 2011.

[21] P. D.-I. J. Gausemeier. Thinking ahead the future of additive manufacturing – analysis of promising industries, 2011. URL https://dmrc.uni-paderborn.de/fileadmin/dmrc/06_Downloads/01_Studies/DMRC_Study_Part_1.pdf.

[22] A. T. Gaynor and J. K. Guest. Topology optimization for additive manufacturing: Considering maximum overhang constraint. *Multidisciplinary Analysis and Optimization Conference*, 2014.

[23] A. T. Gaynor and J. K. Guest. Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design. *Struct Multidisc Optim*, 2016.

[24] A. Gersborg-Hansen. *Topology optimization of flow problems*. PhD thesis, Technical University of Denmark, 2007.

[25] A. Gersborg-Hansen, O. Sigmun, and R. Haber. Topology optimization of channel flow problems. *Structural Multidisc*, 2005.

[26] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 2000.

[27] C. J. Greenshields. *OpenFOAM Programmer's Guide*. OpenFOAM Foundation Ltd., 2015.

[28] C. J. Greenshields. *OpenFOAM User's Guide*. OpenFOAM Foundation Ltd., 2015.

[29] J. Guest and J. Prévost. Topology optimization of creeping fluid flows using a darcy-stokes finite element. *International journal for numerical methods in engineering*, 2006.

[30] E. Helgason. *Development of adjoint-based optimization methods for ducted flows in vehicles*. PhD thesis, Chalmers University of Technology, 2015.

[31] O. Higounenc. Correlation of shot peening parameters to surface characteristic. *ICSP9*, 2005.

[32] C. Hinterberger and M. Olesen. Industrial application of continuous adjoint flow solvers for the optimization of automotive exhaust systems. *CFD & Optimization*, 2011.

[33] M. Hovilehto. Characaterization of design of a product for additive manufacturing. Master's thesis, LAPPEENRANTA UNIVERSITY OF TECHNOLOGY, 2016.

[34] B. J, A. Grace, J. J, and V. G. A review of different etching methodologies and impact of various etchants in wet etching in micro fabrication. *International Journal of Innovative Research in Science, Engineering and Technology*, 2014.

[35] S. G. Johnson. Notes on adjoint methods. URL http://math.mit.edu/~stevenj/18.336/adjoint.pdf.

[36] M. Kaufmann. Cost/weight optimization of aircraft structures. Master's thesis, KTH School of Engineering Sciences, 2008.

[37] K. Kempen, F. Welkenhuyzen, and J.-P. Kruth. Dimensional accuracy of internal channels in slm produced parts, 2015.

[38] S. Kreissl. *Topology Optimization of Flow Problems Modeled by the Incompressible Navier-Stokes Equations*. PhD thesis, Technische Universitat Munchen, 2007.

[39] S. Kreissl, G. Pingen, and K. Maute. Topology optimization for unsteady flow. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, 2011.

[40] S. Kreissl, G. Pingen, and K. Maute. An explicit level set approach for genegeneral shape optimization of fluids with the lattice boltzmann method. *International Journal for Numerical Methods in Fluids*, 2011.

[41] M. Kuron. 3 criteria for assessing cfd convergence. 2015.

[42] M. Langelaar. Topology optimization of 3d self-supporting structures for additive manufacturing. *Additive Manufacturing*, 2016.

[43] H. P. Langtangen and K.-A. Mardal. Numerical methods for incompressible viscous flow. URL https://wiki.math.ntnu.no/_media/ma8502/2014h/langtangen_etal_awr25.pdf.

[44] A. L. Li Yang, Hengfeng Gu. Surface treatment of ti6al4v parts made by powder bed fusion additive manufacturing processes.

[45] G. Liu. Discrete adjoint sensitivity analysis for fluid flow topology optimization based on the generalized lattice boltzmann method. *Computers and Mathematics with Applications*, 2014.

[46] J. M. McDonough. Introductory lectures on turbulence. URL https://www.engr.uky.edu/~acfd/lctr-notes634.pdf.

[47] M. J. Nel. G, Dirker J. Two-dimensional topology optimization of fluid channel distributions - pressure objective. *International Conference on Heat Transfer*, 2014.

[48] U. Nilsson. Description of adjointshapeoptimizatadjoint and how to implement new objective functions. Chalmers University Course. URL http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2013/UlfNilsson/reportAdjoint.pdf.

[49] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[50] C. Online. Turbulence intensity. Internet. URL https://www.cfd-online.com/Wiki/Turbulence_intensity.

[51] C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS*, 2008.

[52] C. Othmer, E. de Villiers, and H. G. Weller. Implementation of a continuous adjoint for topology optimization of ducted flows. *Computational Fluid Dynamics Conference*, 2007.

[53] J. Overvelde. Learn abaqus script in one hour, 2010. URL http://bertoldi.seas.harvard.edu/files/bertoldi/files/learnabaqusscriptinonehour.pdf.

[54] J. Pakkanen, F. Calignano, F. Trevisan, M. Lorusso, E. P. Ambrosio, D. Manfredi, and P. Fino. Study of internal channel surface roughnesses manufactured by selective laser melting in aluminum and titanium alloys. *Metallurgical and Materials Transactions A*, 2016. doi: doi:10.1007/s11661-016-3478-7.

[55] E. Papoutsis-Kiachagias and K. Giannakoglou. Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Computational Methods Engineering*, 2016.

[56] A. Pereira, C. Talischi, G. H. Paulino, I. F. M. Menezes, and M. S. Carvalho. Fluid flow topology optimization in polytop: stability and computational implementation. *Struct Multidisc Optim*, 2013.

[57] A. Pereira, C. Talischi, and G. H. Paulino. Fluid flow topology optimization in polytop: stability and computational implementation. *Structural Multidisciplinary Optimization*, 2016.

[58] B. Philippi and Y. Jin. Topology optimization of turbulent fluid flow with a sensitive porosity adjoint method (spam). *x*, x.

[59] R. A. Pielke. The rise and fall of the space shuttle. *The Scientific Research Societ*, 2008.

[60] M. Pietropaoli, R. Ahlfeld, F. Montomoli, A. Ciani, and M. D'Ercole. Design for additive manufacturing: Internal channel optimization. *Journal of Engineering for Gas Turbines and Power*, 2017.

[61] G. Pyka, G. Kerckhofs, I. Papantoniou, M. Speirs, , J. Schrooten, and M. Wevers. Surface roughness and morphology customization of additive manufactured open porous ti6al4v structures. *Materials*, 2013.

[62] X. Qian. Topology optimization for additive manufacturing: Considering support structures through projected undercut perimeter. *XXIV ICTAM*, 2016.

[63] X. Qian. Undercut and overhang angle control in topology optimization: A density gradient based integral approach. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, 2017.

[64] G. Recktenwald. The simple algorithm for pressure-velocity coupling. URL http://web.cecs.pdx.edu/~gerry/class/ME448/notes_2012/pdf/SIMPLEslides_2up.pdf.

[65] *Additive manufacturing: opportunities and constraints*. Royal Academy of Engineering, 2013.

[66] F. Russoa and N. T. Basse. Scaling of turbulence intensity for low-speed flow in smooth pipes. *Flow Measurement and Instrumentation*, 2016.

[67] T. Saad. Turbulence modeling for beginners. URL https://www.cfd-online.com/W/images/3/31/Turbulence_Modeling_For_Beginners.pdf.

[68] M. R. Sankar, V. K. Jain, and J. Ramkumar. Abrasive flow machining (afm): An overview. *Researchgate*, 2014.

[69] M. R. Serphos. Incorporating am-specific manufacturing constraints into topology optimization. Master's thesis, TU Delft, 2014.

[70] O. Sigmund. A 99 line topology optimization code written in matlab. *Structural Multidisciplinary Optimization*, 2001.

[71] J. C. Snyder, C. K. Stimpson, and K. A. Thole. Build direction effects on additively manufactured channels. *Turbine Technical Conference and Exposition*, 2015.

[72] G. P. Steven, Q. Li, and Y. M. Xie. Evolutionary topology and shape design for general physical field problems. *Computational Mechanics*, 2000.

[73] T. Tatiana and N. Alexey. Technological limitations of selective laser melting method. *Applied Mechanics and Materials*, 2014.

[74] D. Thomas. *The Development of Design Rules for Selective Laser Melting*. PhD thesis, University of Wales Institute, 2009.

[75] K. Tosha. Characteristics of shot peened surfaces and surface layers. *Asia-Pacific Forum on Precision Surface Finishing and Deburring Technology*, 2001.

[76] M. Towara and U. Naumann. A discrete adjoint model for openfoam. 2013.

[77] H. Versteeg and W.Malalasekera. *An introduction to Computational Fluid Dynamics*. Longman Scientific & Techinical, 1995.

[78] F. M. White. *Fluid Mechanics*. McGraw-Hill, 2011.

[79] S. Zhou and Q. Li. A variational level set method for the topology optimization of steady-state navier–stokes flow. *Journal of Computational Physics*, 2008.